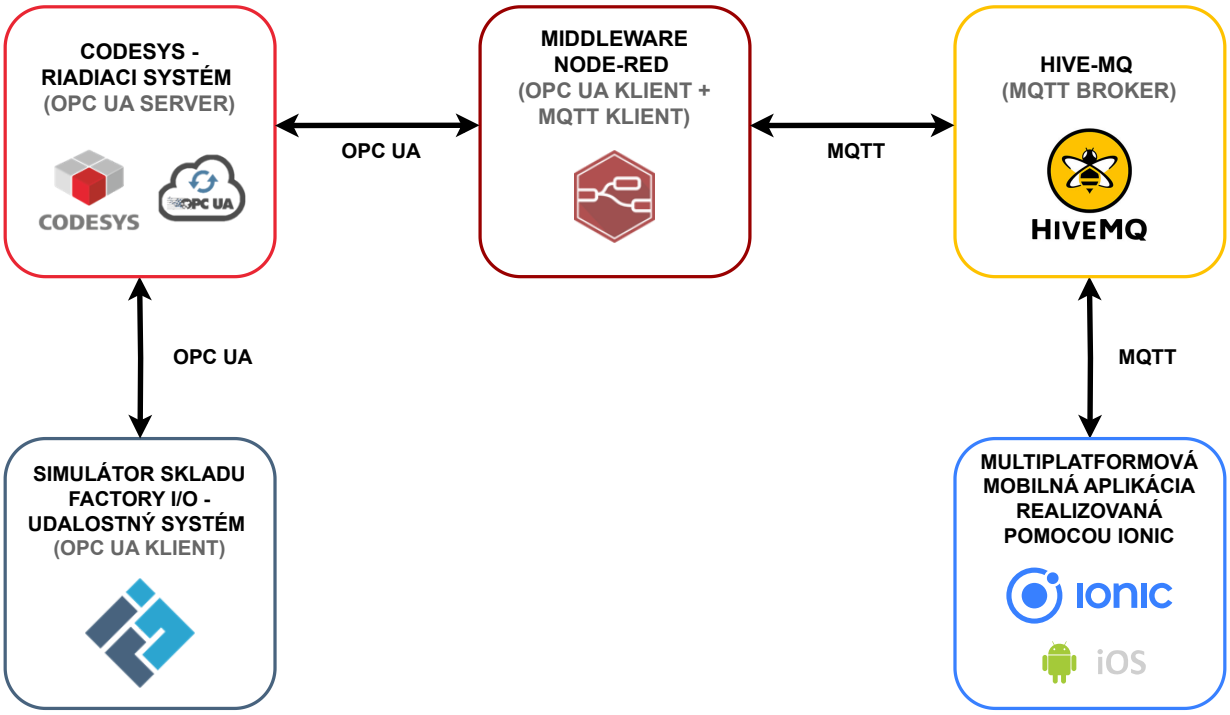
4.4 Štvrtá prípadová štúdia: Prepojenie PLC s mobilnou aplikáciou

V tejto edukačnej prípadovej štúdii je uvedené riešenie pre automatizáciu virtuálneho skladového systému, ktorý je riadený prostredníctvom softPLC (realizovanom v Codesys) a mobilnej aplikácie. Virtuálny skladový systém bol realizovaný vo Factory I/O, ktorý bude figurovať ako OPC UA klient. Na prepojenie mobilnej aplikácie s virtuálnym skladovým systémom je využívaný middleware Node-RED. Node-RED bude figurovať ako OPC UA klient. S mobilnou aplikáciou bude Node-RED komunikovať cez MQTT protokol. Táto prípadová štúdia je opísaná s využitím diplomovej práce [76].

Architektúra celého systému je navrhnutá na základe technológií, ktoré sú ľahko dostupné. Návrh je znázornený na obrázku č. 136.



Obrázok č. 136: Architektúra aplikačného systému štvrtej prípadovej štúdie

Medzi tri hlavné komponenty patrí virtuálny skladový systém, softPLC a mobilná aplikácia. Na počítači s operačným systémom Windows máme nainštalovaný softvér Codesys a simulátor Factory I/O. Factory I/O poskytuje vopred pripravenú scénu so skladovým systémom, ktorý máme za úlohu riadiť. Na riadenie skladového systému využívame Codesys, ktorý okrem vývojového prostredia pre PLC program zabezpečuje aj softPLC bežiace na počítači. Codesys runtime (figuruje ako OPC UA server) a Factory I/O budú komunikovať prostredníctvom OPC UA protokolu modelom klient-server.

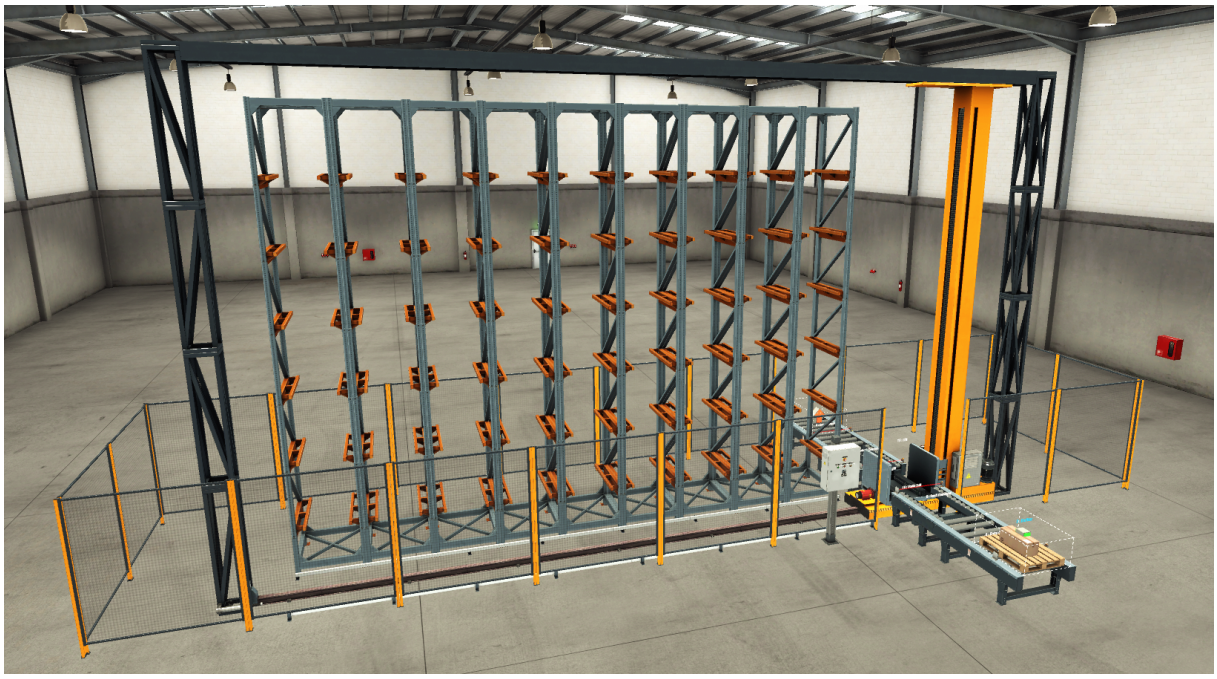
Aby sme mohli do tejto zostavy pridať aj mobilnú aplikáciu, využívame middleware

Node-RED. V Node-REDe máme ďalšieho OPC UA klienta, ktorý sa napája na OPC UA server. Dáta medzi mobilnou aplikáciou a Node-REDom sa vymieňajú prostredníctvom protokolu MQTT. Využije sa HiveMQ, čo je MQTT broker ktorý spracúva údaje a posiela ich ďalej k daným MQTT klientom.

4.4.1 Špecifikácia a správanie diskrétneho udalostného systému

Využívaným diskrétnym udalostným systémom je virtuálny automatizovaný skladový systém od tvorcov softvéru Factory I/O (obrázok č. 137).

Model pozostáva z elektrického rozvádzača s tlačidlami, vysielača (angl. emitter), dopravníkového pásu (ide o model s valcami pre ťažšie predmety), nakladacieho pásu, retroreflexných senzorov, aktuátorov, koľajnicového stohovacieho žeriavu a skladu. Koľajnicový stohovací žeriav obsahuje vozík, vertikálnu platformu, dve vidlice a skladové miesto.



Obrázok č. 137: 3D model skladového systému [76]

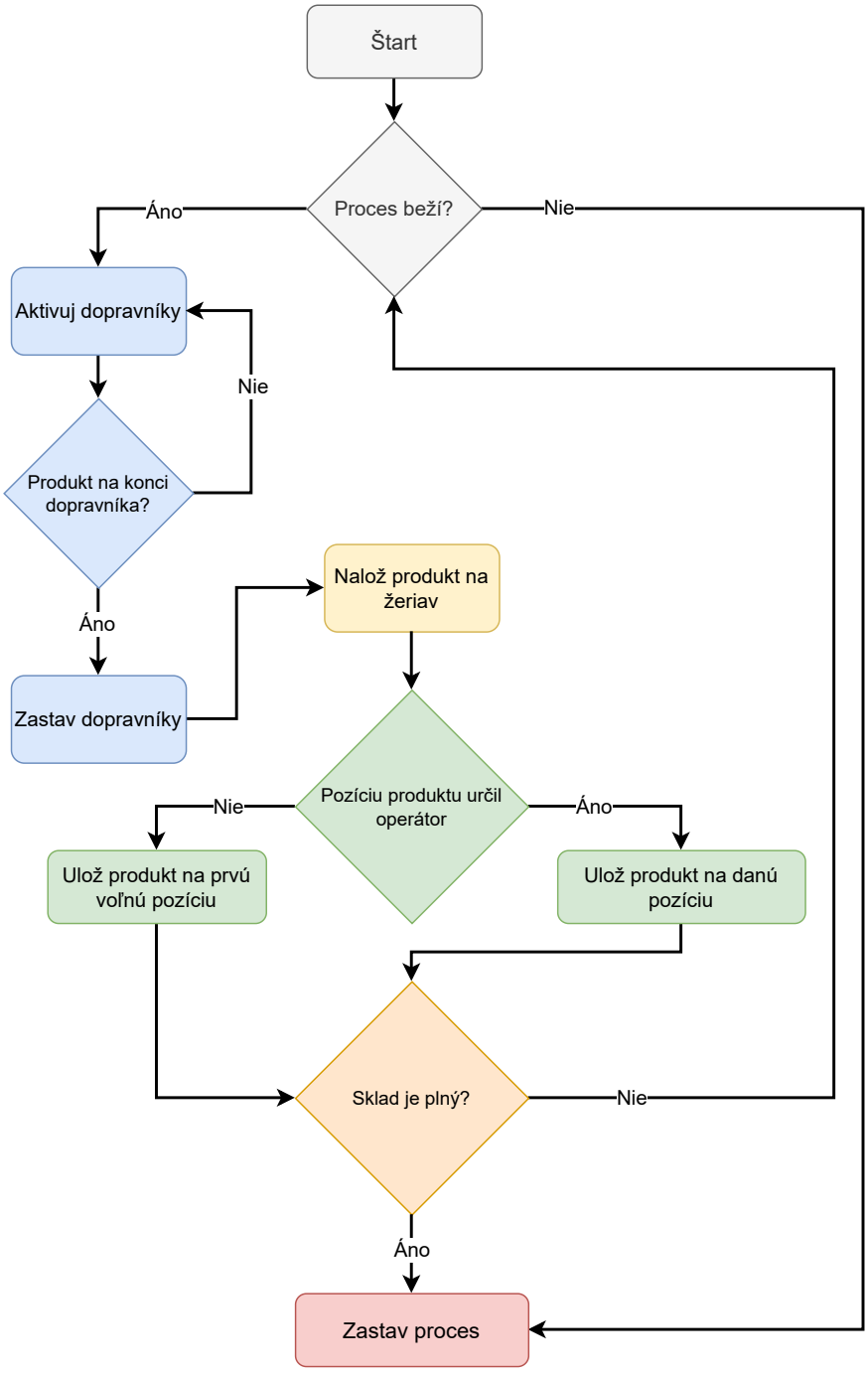
Na vertikálnej platforme a na vozíku sa nachádzajú dva laserové diaľkomery, ktoré merajú horizontálnu a vertikálnu polohu platformy. Stohovací žeriav je možné ovládať digitálnymi, numerickými a analógovými hodnotami podľa zvolenej konfigurácie. Pre jednoduchosť a názornosť sú v našom prípade nastavené numerické hodnoty. Polohu žeriavu určuje premenná s názvom *TargetPosition*. Jeho začiatočná hodnota je číslo 55. Celkový počet buniek v sklade je 54. Toto znamená, že cieľová bunka v sklade a zároveň aj poloha žeriavu môže byť definovaná celočíselnou hodnotou medzi 1 a 54. Ak je táto hodnota nastavená na nulu, stohovací žeriav sa zastaví v aktuálnej polohe. Ak je však

vyššia ako 54, žeriav sa vráti do začiatočnej polohy, teda na číslo 55.

Cieľom riadenia je pomocou spomenutých dielov výrobok umiestniť do skladu. Proces začína vygenerovaním výrobku z vysielača (emitter). Akonáhle sa produkt dostaví na pás, retroreflexný senzor ho rozpozná a čaká sa na operátora, aby zatlačil tlačidlo Štart. Stlačením tohto tlačidla sa aktivuje valcový pás a produkt sa posunie k nakladaciemu pásu. Na konci nakladacieho pásu je ďalší senzor, ktorý keď rozpozná výrobok, tak zastaví oba pásy a nasleduje naloženie výrobku na žeriav. Akonáhle žeriav naloží výrobok do vidlice, nasleduje transport výrobku do skladu. Ako už bolo spomenuté, žeriav je v tomto prípade ovládaný numerickými hodnotami. Do premennej aktuátora *TargetPosition* zašleme hodnotu z intervalu 1 až 54 a týmto určíme polohu žeriavu a pozíciu výrobku v sklade.

Jednou z požiadaviek tejto časti riadenia je, aby výber skladového miesta bol automatický alebo manuálny. Pokiaľ operátor neurčí pozíciu produktu v sklade sám, program, pomocou ktorého riadime diskrétny udalostný systém, automatický určí prvú voľne dostupnú pozíciu v sklade a tam výrobok umiestni. Keď žeriav produkt uskladní, vráti sa späť na začiatočnú polohu. Tento proces opakujeme, pokiaľ sa všetky bunky v sklade neobsadia alebo pokiaľ operátor nezatlačí tlačidlo Stop.

Pre lepšiu názornosť sme správanie skladového systému vyjadrili pomocou vývojového diagramu — obrázok č. 138.

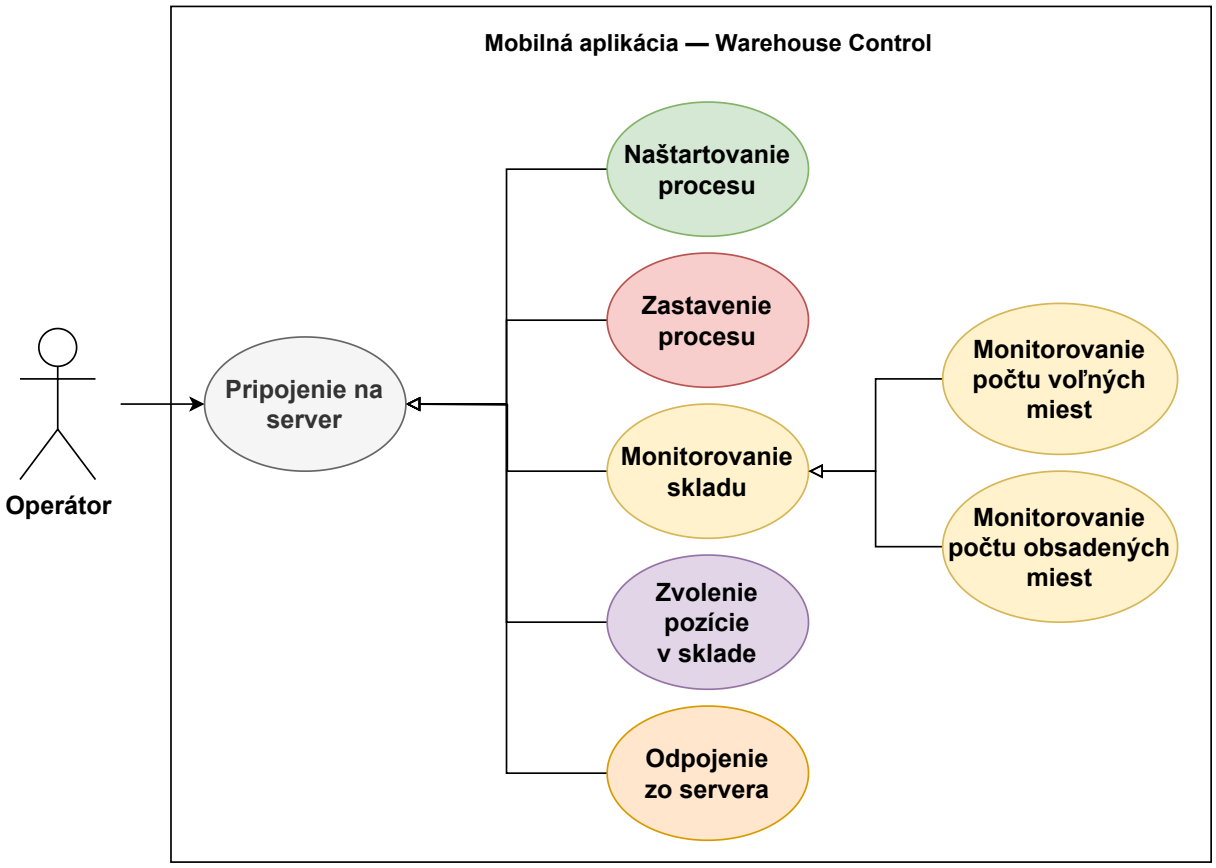


Obrázok č. 138: Vývojový diagram skladového systému [76]

4.4.2 Špecifikácia mobilnej aplikácie

Ako už bolo spomínané, operátor riadi a monitoruje sklad s využitím mobilnej aplikácie. To znamená, že v mobilnej aplikácii je potrebné mať možnosť naštartovania procesu stlačením tlačidla Štart, zastavením procesu stlačením tlačidla Stop a proces celkom ukončiť odpojením sa zo servera. Taktiež má operátor možnosť monitorovať počet voľných miest v sklade, ako aj počet produktov, ktoré sa v sklade nachádzajú. Poslednou požiadavkou je, aby mal možnosť si vybrať, na akú pozíciu chce produkt do skladu

umiestniť. Diagram prípadov použitia, ktorý je znázornený na obrázku č. 139, opisuje všetky požiadavky týkajúce sa mobilnej aplikácie.

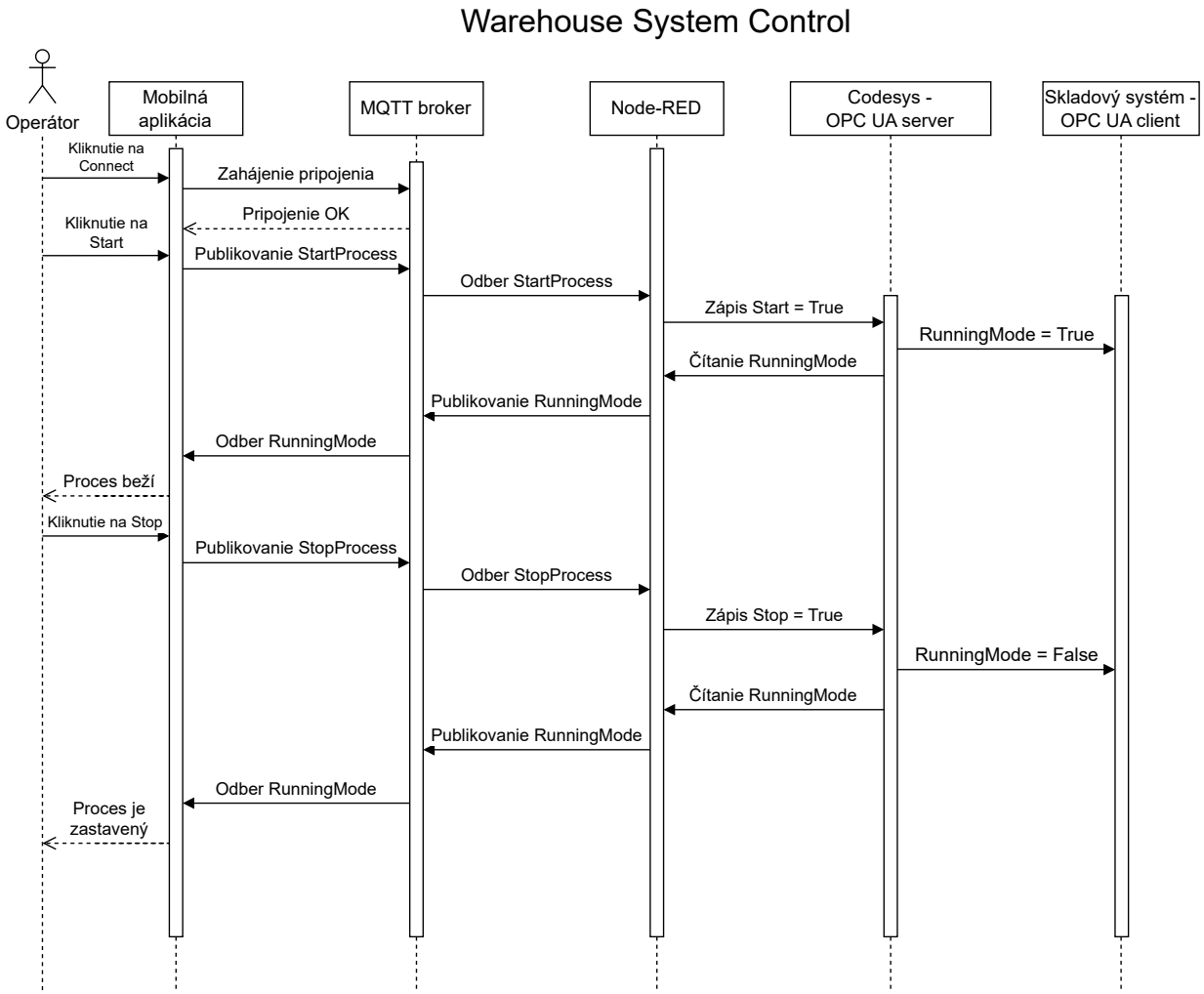


Obrázok č. 139: Diagram prípadov použitia mobilnej aplikácie [76]

4.4.3 Komunikácia jednotlivých subsystémov

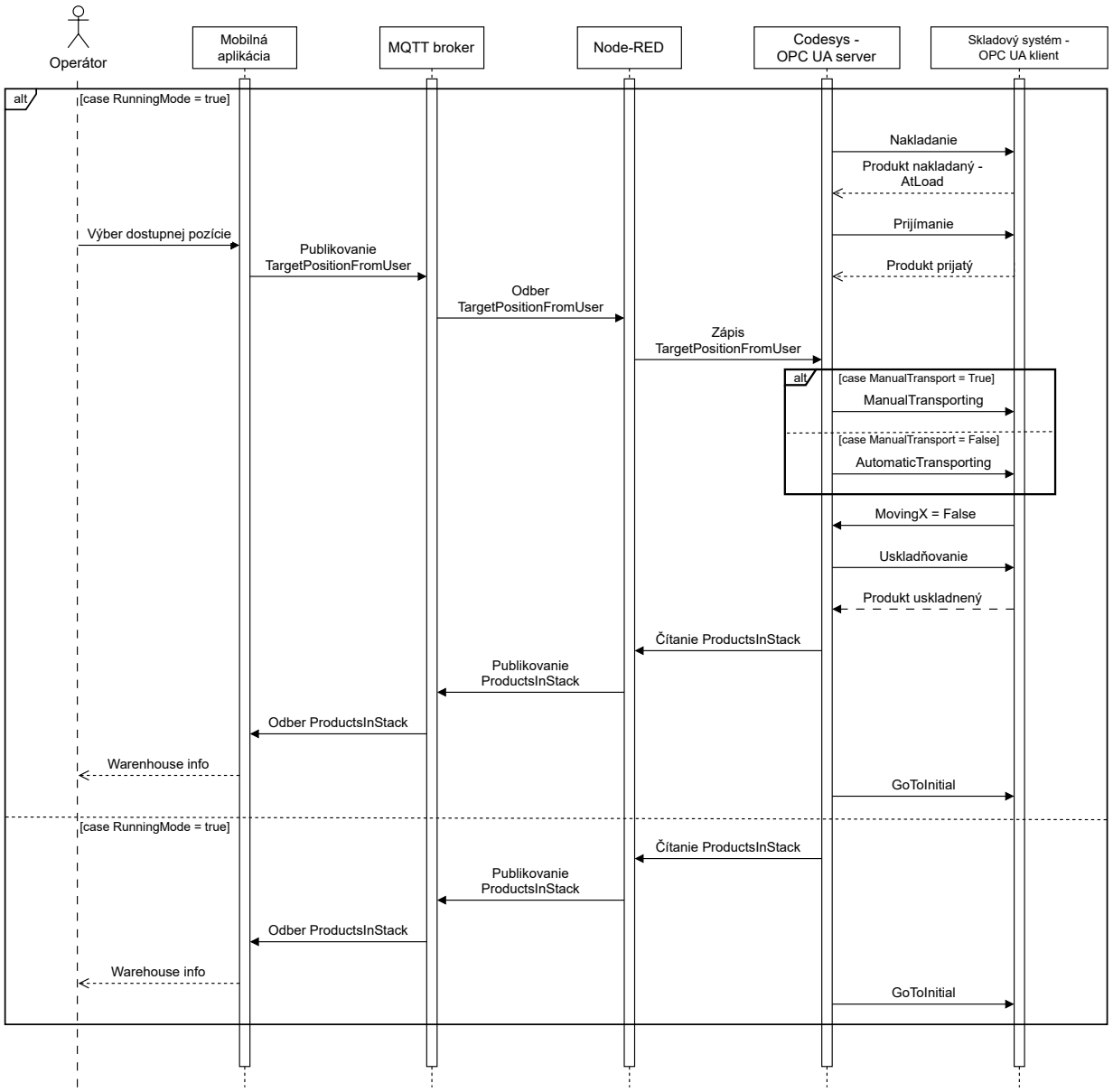
Veľmi podstatným prvkom v celom systéme je komunikácia. Na základe dostupných technológií a možností sme navrhli také riešenie, ktoré zabezpečí rýchly prenos údajov medzi systémami a umožní operátorovi sledovať aktuálny stav skladových priestorov.

Proces začína od operátora, ktorý sa najprv pripojí na MQTT broker HiveMQ. Akonáhle sa mu to podarí, má možnosť naštartovať proces zatlačením tlačidla Start. Hodnota z tlačidla sa najprv zašle na MQTT broker a MQTT klient v Node-RED túto zmenu zachytí tým, že vykoná subscribe na danú tému (*topic*), pod ktorou je hodnota zapísaná/vysielaná. Následne sa táto hodnota prepošle prostredníctvom OPC UA klienta v Node-REDe na OPC UA server v Codesyse. Akonáhle OPC UA server v Codesyse túto hodnotu zapíše, prepošle ju ďalej k OPC UA klientovi vo Factory I/O. Táto časť komunikácie je znázornená na obrázku č. 140.



Obrázok č. 140: Sekvenčný diagram — prípad č.1 [76]

Na obrázku č. 141 je znázornená komunikácia v dvoch prípadoch: keď proces beží a keď je zastavený. Ak proces beží, operátor má možnosť vybrať si pozíciu v sklade, kam chce výrobok uskladniť. Akonáhle skladový systém vo Factory I/O tento výrobok uskladní, zašle o tom informáciu späť operátorovi. V prípade, že proces nebeží, operátor vidí iba to, aký je aktuálny stav v sklade a má možnosť proces opäť naštartovať.



Obrázok č. 141: Sekvenčný diagram — prípad č.2 [76]

4.4.4 Riadenie diskrétneho udalostného systému

Riadenie skladového systému prebieha prostredníctvom softPLC v Codesyse. Ako už bolo spomenuté, Codesys je softvérový systém zložený z dvoch častí: integrované vývojové prostredie (IDE) a runtime. Integrované vývojové prostredie sme využili na implementáciu programu skladového systému.

Codesys Control Win (runtime) poskytuje virtuálne softPLC, ktoré beží na lokálnom OPC UA serveri. Beh programu sa na každé 2 hodiny vypne a je potrebné ho znovu naštartovať, čo je spôsobené licenciou, ktorá je v tomto móde zadarmo.

Z vývojového diagramu na obrázku č. 138 je možné vidieť, že celý proces skladovania je rozdelený do niekoľkých krokov. Jednotlivé kroky v procese sčasti závisia od operátora. Z tohto dôvodu riadenie členíme na **manuálne** a **automatické**. Manuálne

v tomto prípade znamená, že operátor môže naštartovať proces, ukončiť a vybrať si miesto v sklade, kam chce výrobok uskladniť.

Aplikácia v Codesyse je zložená z objektov rôznych typov. Zdrojový kód pre program v PLC je objekt typu *programovacia organizačná zložka* (*POU*). Nakoľko opisovaný proces čiastočne závisí od operátora (operátor je ten, ktorý proces naštartuje stlačením tlačidla Štart), boli vytvorené dva programy (dve POU): *Control* a *Main*.

Control POU je pomocný program, ktorý beží neustále popri hlavnom programe na ovládanie vykonávania sekvencií hlavného programu na základe vstupu od operátora.

Main POU je hlavný program, ktorý je zložený z niekoľkých krokov, ktoré riadia proces v sklade. Aby všetko toto bolo možné implementovať, prvým krokom bolo definovanie a inicializácia globálnych premenných.

Globálne premenné v PLC sú bežné premenné, ktoré sú rozpoznané v rámci celého projektu, teda je možné k ním pristupovať, čítať a meniť ich hodnoty odkiaľkoľvek v projekte. V rámci projektu boli vytvorené dva zoznamy globálnych premenných: **FIO** a **ControlPou**. Zoznam s názvom FIO obsahuje všetky globálne premenné, ktoré reprezentujú senzory a aktuátory v opisovanom skladovom systéme (obrázok č. 142). Zoznam premenných s názvom ControlPou (obrázok č. 143) obsahuje globálne premenné, ktoré budú využívané na prepájanie medzi manuálnym a automatickým riadením a taktiež na vizualizáciu aktuálneho stavu procesu v mobilnej aplikácii.

Globálna premenná *RunningMode* v zozname ControlPou je boolovského typu, ktorej hodnota bude spúšťať a zastavovať proces v hlavnom programe. Premenná s názvom *ManualTransport* je tiež boolovského typu, ktorej hodnota prepája riadenie medzi manuálnym a automatickým v hlavnom programe. Premenná *TargetPositionFromUser* ukladá hodnotu, ktorú operátor zašle prostredníctvom mobilnej aplikácie, keď chce určiť pozíciu výrobku v sklade. Je dátového typu WORD, čo je 16-bitové celé číslo bez znamienka (UInt16). Premenná *ProductsInStack* je pole veľkosti 54, ktoré reprezentuje sklad, presnejšie skladové miesta. Taktiež je dátového typu WORD. Na začiatku je pole vyplnené nulami. Nuly reprezentujú prázdne miesta v sklade. Akonáhle sa nejaký výrobok uskladní do skladu, jeho pozíciu zapíšeme do tohto poľa. Využívame ho z dôvodu, aby operátor mohol v mobilnej aplikácii vidieť, ktoré miesta v sklade sú voľné a ktoré sú už obsadené. A taktiež v prípade, keď operátor neurčí danú pozíciu, aby program automaticky vyhľadal prvé voľné miesto a tam výrobok uskladnil.

FIO

```
VAR_GLOBAL
    //sensors
    AtEntry:BOOL;
    AtExit:BOOL;
    AtLeft:BOOL;
    AtLoad:BOOL;
    AtMiddle:BOOL;
    AtRight:BOOL;
    AtUnload:BOOL;
    Auto:BOOL;
    EmergencyStop:BOOL;
    Maual:BOOL;
    MovingX:BOOL;
    MovingZ:BOOL;
    Reset:BOOL;
    Start:BOOL;
    Stop:BOOL;

    //actuators
    EntryConveyor:BOOL:=FALSE;
    ExitConveyor:BOOL:=FALSE;
    ForksLeft:BOOL:=FALSE;
    ForksRight:BOOL:=FALSE;
    Lift:BOOL:=FALSE;
    LoadConveyor:BOOL:=FALSE;
    ResetLight:BOOL:=FALSE;
    StartLight:BOOL:=FALSE;
    StopLight:BOOL:=FALSE;
    TargetPosition:WORD:=0;
    UnloadConveyor:BOOL:=FALSE;
END_VAR
```

Obrázok č. 142: Zoznam globálnych premenných pre riadenie skladového systému vo Factory I/O [76]

ControlPou

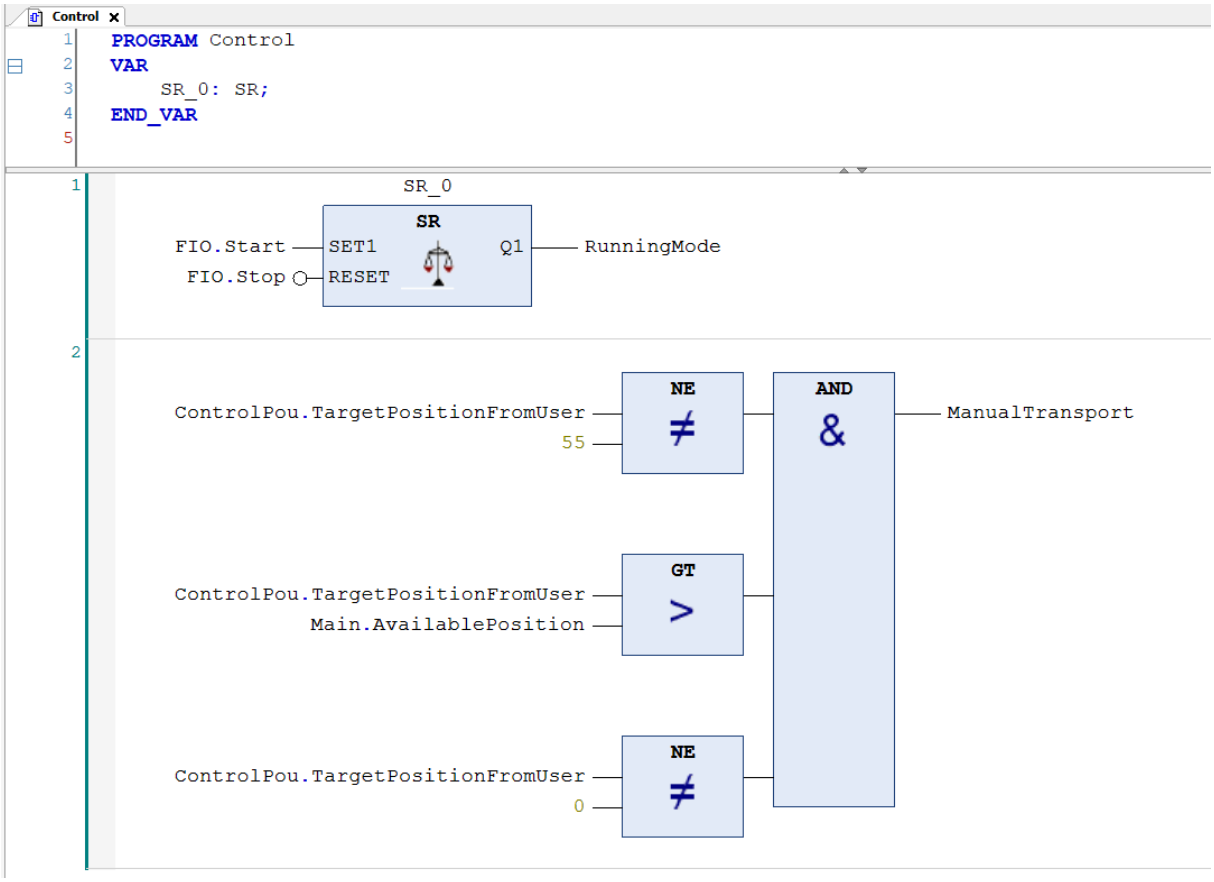
```
VAR_GLOBAL
    RunningMode: BOOL;
    ManualTransport: BOOL;
    //additional variable to control position
    TargetPositionFromUser: WORD := 0;
    ProductsInStack: ARRAY[0..53] OF WORD;
END_VAR
```

Obrázok č. 143: Zoznam globálnych premenných pre prepojenie medzi manuálnym a automatickým riadením [76]

POMOCNÝ PROGRAM — CONTROL

Na implementáciu pomocného programu Control (obrázok č. 144) sme využili programovací jazyk Function Block Diagram (FBD), nakoľko tu riadime dva procesy, ktoré závisia iba od vstupu od operátora, čo sa prehľadne dá v tomto jazyku znázorniť.

V prvom bloku programu je nastavovaná hodnota globálnej premennej *RunningMode*, ktorá je boolovského typu. Jej hodnota sa nastaví na základe hodnoty z tlačidla Štart alebo Stop. Na nastavenie hodnoty používame **SR blok**. SR blok má dva vstupy Set a Reset a výstup Q. Aktivácia výstupu Q je určená hodnotou zo vstupu Set, čo je v opisovanom prípade hodnota tlačidla Start. Deaktivácia výstupu je určená hodnotou zo vstupu Reset, čo je hodnota z tlačidla Stop. Tu ale bolo nutné do vstupu Reset poslať negovanú hodnotu, lebo predvolená hodnota tlačidla Stop je TRUE. Pod aktiváciou a deaktiváciou výstupu je možné rozumieť priradenie hodnôt TRUE a FALSE do premennej *RunningMode*. Hodnota z premennej *RunningMode* následne štartuje a zastavuje proces v hlavnom programe.



Obrázok č. 144: Pomocný program Control [76]

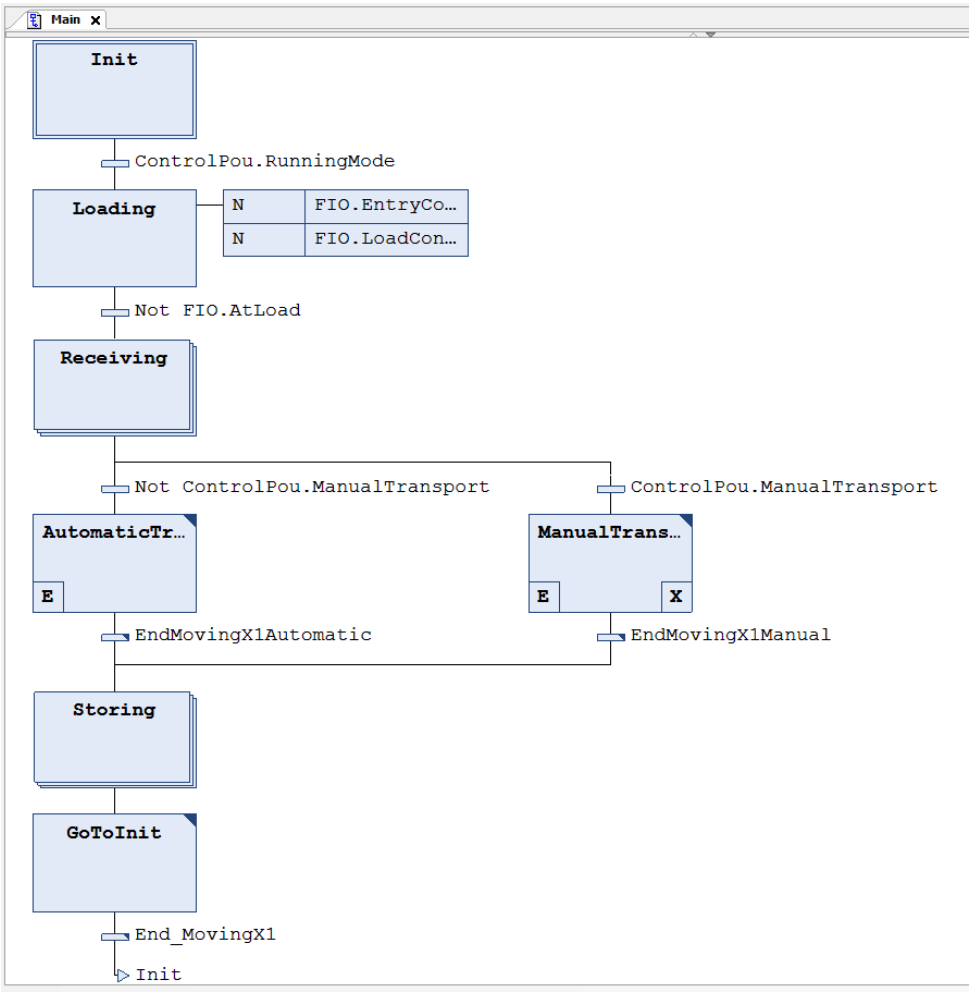
V druhom bloku programu je nastavovaná hodnota globálnej premennej *ManualTransport*, ktorá je boolovského typu. Na nastavenie hodnoty je využívaný boolovský

operátor AND s tromi vstupmi. Vstupy sú matematické operátory. Dva sú pre porovnanie nerovnosti a jeden operátor porovnania „väčší ako“. Tu ošetrujeme hodnoty, ktoré poslal užívateľ z mobilnej aplikácie a ak všetky operátory spĺňajú podmienku (teda majú výstupnú hodnotu TRUE) globálna premenná *ManualTransport* nadobudne hodnotu TRUE. V opačnom prípade jej hodnota bude FALSE. Pomocou tejto premennej sa bude ovládať prepínanie medzi automatickým a manuálnym riadením v hlavnom programe.

HLAVNÝ PROGRAM — MAIN

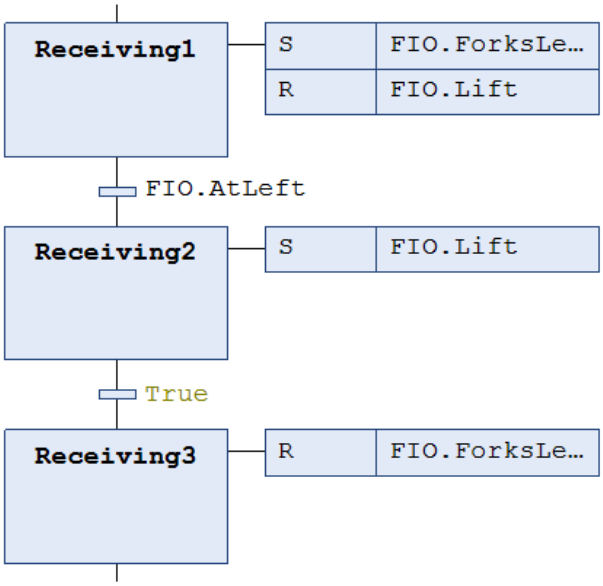
Riadenie hlavného programu možno rozdeliť do niekoľkých krokov. Tieto kroky predstavujú dané procesy v skladovom systéme, ktoré sa vykonávajú postupne jeden za druhým. Celý proces v skladovom systéme sa teda vykonáva sekvenčne. Z tohto dôvodu sme sa rozhodli na implementáciu programu využiť programovací jazyk Sekvenčný funkčný diagram (SFC).

Program sa skladá zo 7 blokov (obrázok č. 145).



Obrázok č. 145: Hlavný program Main [76]

Na začiatku je prítomný inicializačný blok *Init*, ktorý prejde do ďalšieho kroku, keď hodnota globálnej premennej *RunningMode* bude TRUE. V ďalšom kroku je prítomný blok *Loading*, ktorý aktivuje valcový a výrobný pás. Na aktiváciu pásov boli využité dve IEC akcie. Tieto akcie sa vykonajú iba, keď je blok *Loading* aktívny. V tomto prípade to znamená, že sa blok deaktivuje a obidva pásy sa zastavia, keď výrobok príde po koniec nakladacieho pásu, kde ho retroreflexný senzor *AtLoad* zdeteguje. Nasledovným krokom je blok *Receiving*. Tu bolo využité makro a jednotlivé bloky sme zabalili, nakoľko sa krok pre naloženie výrobku na žeriav skladá z troch častí (obrázok č. 146).



Obrázok č. 146: Makro pre Receiving Step [76]

Najprv sa aktivuje ľavá vidlica žeriavu (aktuátor *ForksLeft*), resp. vysunie sa dopredu, a nakoľko vidlica musí byť pod výrobkom, aktuátor *Lift* musí byť deaktivovaný. Na túto funkcionalitu opäť využívame dve IEC akcie. Akonáhle senzor na žeriave *AtLeft* zdeteguje vysunutie vidlice, prechádza sa do ďalšieho bloku a tu sa iba opäť aktivuje aktuátor *Lift*, aby výrobok zodvihol. Akonáhle je výrobok zdvihnutý, prejde sa do ďalšieho bloku a resetne sa ľavý aktuátor *ForksLeft*. Ďalším krokom je transport výrobku do skladu. Toto bolo vyriešené pomocou alternatívnej vetvy, ktorá obsahuje dva bloky *AutomaticTransporting* a *ManualTransporting*. Iba jeden z nich sa aktivuje, a to v závislosti od toho, aká je hodnota globálnej premennej *ManualTransport*, ktorú nastavujeme v pomocnom programe Control.

Na vstupe do bloku *AutomaticTransporting* je akcia *AutomaticTransport_entry*, ktorá bola implementovaná pomocou štruktúrovaného textu (Programový modul č. 5).

Najprv si je nutné v slučke skontrolovať, či sa hodnota premennej *AvailablePosition*

Programový modul č. 5 Akcia AutomaticTransport_entry [76]

```
1  //Loop to check if AvailablePosition is already in ProductsInStack array
2  REPEAT
3      found:=FALSE;
4      FOR i:=0 TO 53 DO
5          IF ControlPou.ProductsInStack[i] = AvailablePosition THEN
6              found:=TRUE;
7              EXIT;
8          END_IF
9      END_FOR
10     IF found THEN
11         AvailablePosition:=AvailablePosition+1; //If AvailablePosition already exists
12         // in array, increase by 1
13     END_IF
14 UNTIL NOT found
15 END_REPEAT
16
17 //Store AvailablePosition in ProductsInStack array
18 FOR i:=0 TO 53 DO
19     IF ControlPou.ProductsInStack[i] = 0 THEN //CheckForEmptySlotIn ProductsInStack array
20         ControlPou.ProductsInStack[i]:=AvailablePosition;
21         EXIT;
22     END_IF
23 END_FOR
```

už nachádza v poli *ProductsInStack*. Akonáhle je nájdená, premenná *AvailablePosition* sa zvýši o 1, skontroluje sa prvé voľne dostupné miesto v sklade (v poli *ProductsInStack*) a na tú pozíciu sa zapíše hodnota z premennej *AvailablePosition*. Po skončení tejto akcie nasleduje akcia *AutomaticTransport_active*, ktorá je tiež implementovaná pomocou štruktúrovaného textu (Programový modul č. 6).

Programový modul č. 6 Akcia AutomaticTransport [76]

```
1 FIO.TargetPosition := AvailablePosition;
2 ControlPou.ProductsInStack[AvailablePosition - 1] := AvailablePosition;
```

V tejto akcii je najprv do globálnej premennej *TargetPosition* priradená hodnota z premennej *AvailablePosition* a potom táto rovnaká hodnota je priradená aj do poľa *ProductsInStack* na danú pozíciu. Pre ukončenie tohto bloku, ako podmienku pre prechod do nasledovného kroku, máme k dispozícii funkčný blok *F_TRIG*, ktorého hodnota bude TRUE akonáhle sa vozík na žeriave zastaví na danej pozícii (resp. hodnota senzora pohybu *MovingX* bude FALSE).

Na vstupe do bloku *ManualTransporting* sa nachádza akcia *ManualTransport_entry*. Akcia je opäť implementovaná pomocou štrukturovaného textu (programový modul č. 7).

Programový modul č. 7 Akcia *ManualTransport_entry* [76]

```
1 IF ControlPou.ProductsInStack[ControlPou.TargetPositionFromUser - 1] = 0 THEN
2         ControlPou.ProductsInStack[ControlPou.TargetPositionFromUser - 1] :=
3     ControlPou.TargetPositionFromUser;
4 END_IF
```

V tejto akcii ošetríme, či hodnota premennej *TargetPositionFromUser* (teda hodnota, ktorú si zvolil operátor) sa nenachádza v poli *ProductsInStack*. Ak nie, tak ju tam pridáme. Následne sa vykoná akcia *ManualTransport_active*, v ktorej je iba presunutá hodnota z globálnej premennej *TargetPositionFromUser* do aktuátora *TargetPosition*. Na výstupe z bloku máme akciu *ManualTransport_exit*, kde je iba vynulovaná hodnota z globálnej premennej *TargetPositionFromUser*, aby sa resetla hodnota globálnej premennej *ManualTransport* na FALSE. Obidve akcie *ManualTransport_active* a *ManualTransport_exit* obsahujú jeden operátor MOVE na priradenie daných hodnôt.

Opäť, na ukončenie tohto bloku ako podmienku pre prechod do nasledovného kroku je využitý funkčný blok *F_TRIG*.

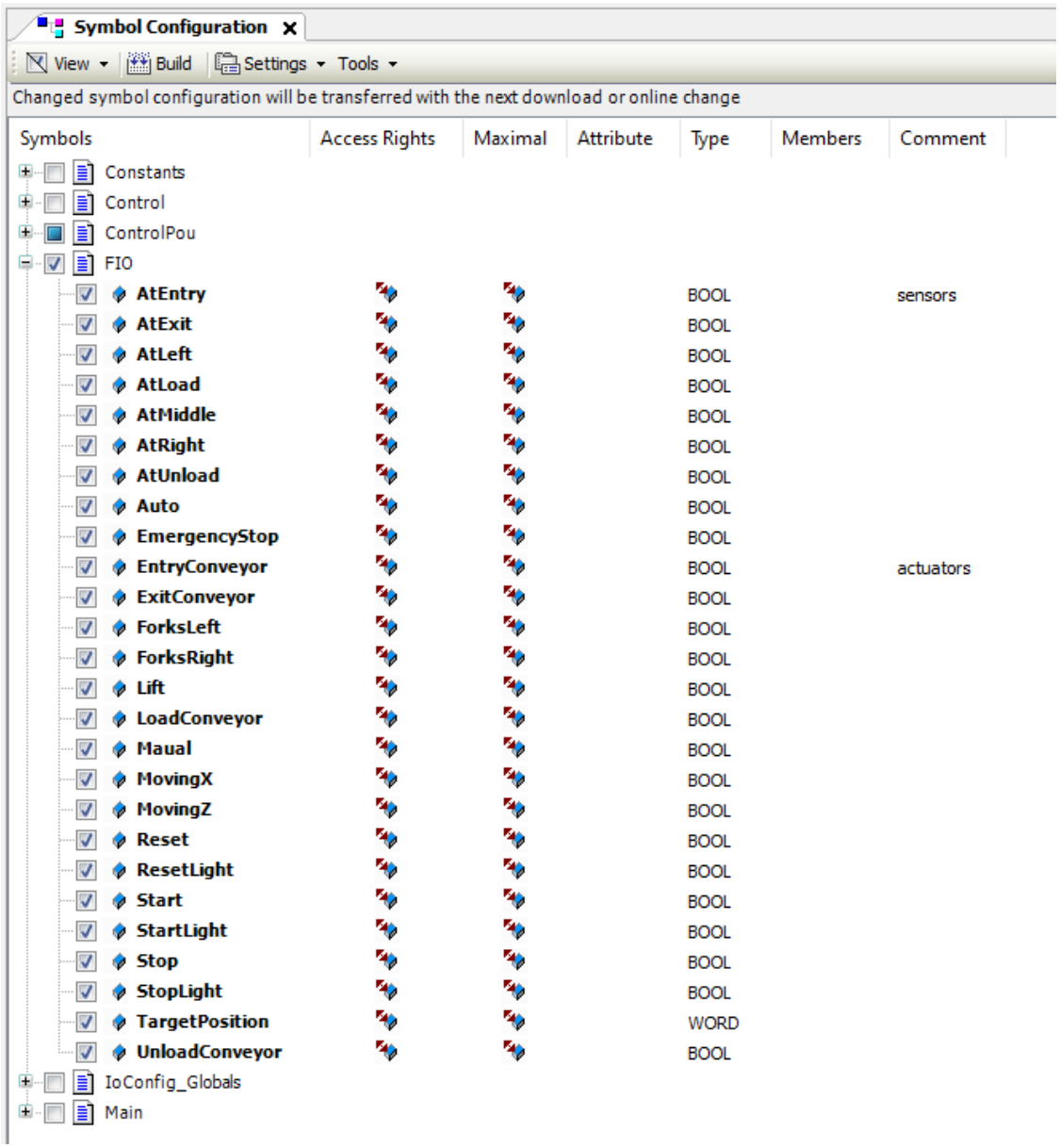
Ako ďalší krok je blok *Storing*, ktorý je inverzný od bloku *Receiving*. Akonáhle je

výrobok vložený do skladu, prejdeme do posledného bloku *GoToInitial*, ktorý má na vstupe akciu *GoToInitial_active*, kde je iba nastavená hodnota globálnej premennej *Target-Position* na 55, aby sa žeriav vrátil späť na začiatočnú pozíciu. Akonáhle sa vozík zastaví (hodnota senzora pohybu *MovingX* bude FALSE) opäť sa aktivuje inicializačný blok *Init* a proces sa opakuje.

4.4.5 Konfigurácia komunikácie medzi softPLC a Factory I/O

Ďalším krokom v implementácii riadenia virtuálneho skladového systému je konfigurácia komunikácie medzi softPLC (Codesys runtime) a Factory I/O. Komunikácia prebieha prostredníctvom protokolu OPC UA, nakoľko oba softvéry podporujú možnosť takejto komunikácie.

Komunikačný model OPC UA v tomto prípade je klient-server. OPC UA klient je Factory I/O a OPC UA server je Codesys runtime. Na konfiguráciu komunikácie je potrebné najprv vytvoriť adresný priestor v OPC UA serveri. Tu je žiadúce, aby adresný priestor obsahoval všetky globálne premenné, ktoré boli zadefinované v zozname FIO (obrázok č. 142). Na tento účel je treba pridať do aplikácie objekt *Symbol Configuration* (obrázok č. 147). Z tohto objektu je nutné vybrať zoznam FIO a označiť všetky premenné. Tieto premenné budú tak dostupné pre OPC UA klienta.

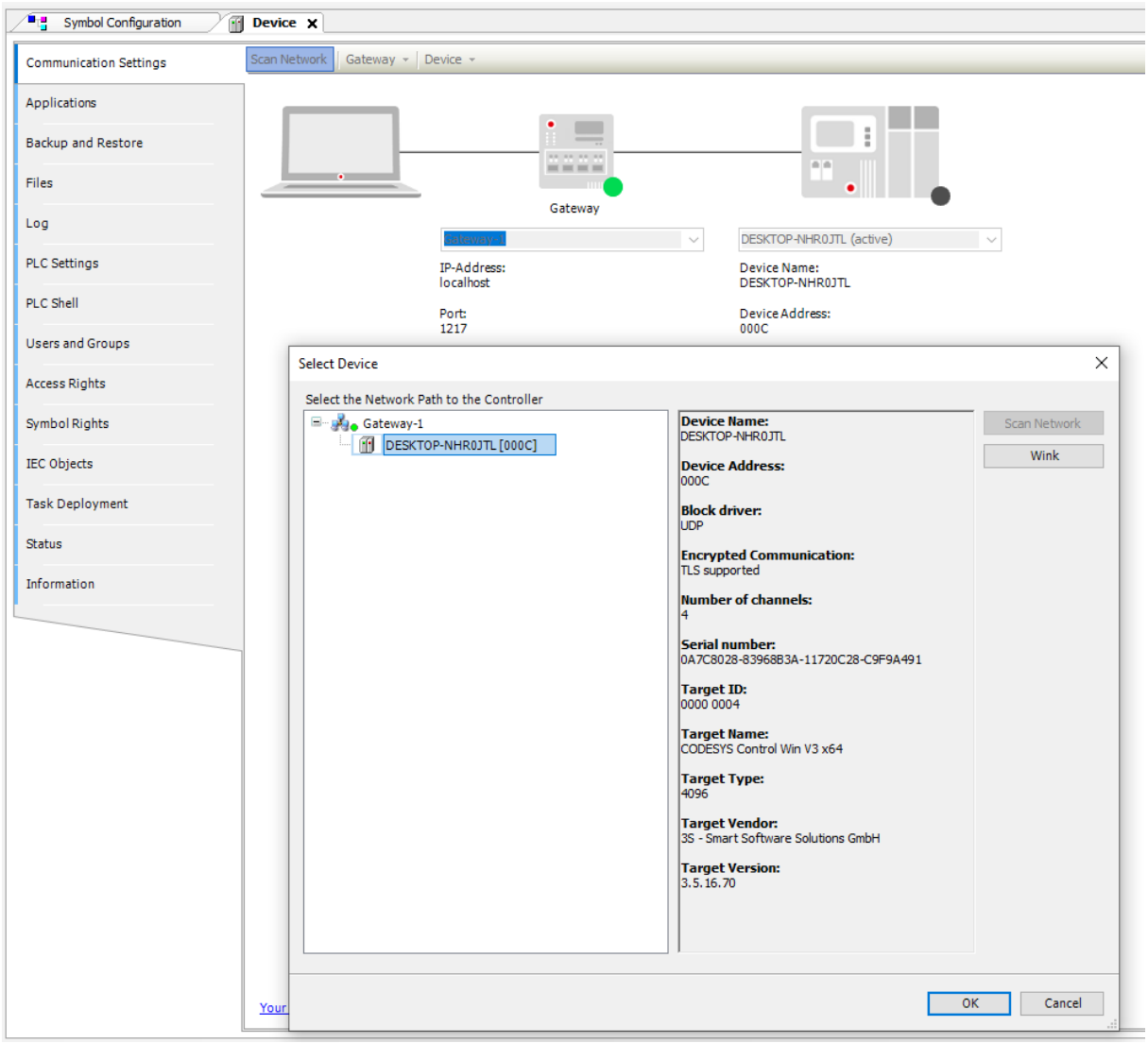


Obrázok č. 147: Objekt Symbol Configuration [76]

Následne nakonfigurujeme zariadenie, čo v tomto prípade bude softPLC. V časti *Communication settings* najprv naskenujeme sieť zatlačením tlačidla *Scan Network* a vyberieme zariadenie z ponuky na obrázku č. 148. Program, ktorý bol vytvorený, nahráme do softPLC zatlačením tlačidla *Build* a projekt následne spustíme.

Ďalším krokom je konfigurácia OPC UA klienta vo Factory I/O. V scéne *Automated Warehouse*, čo je model skladového systému, je potrebné z ponuky ovládačov vybrať *OPC Client DA/UA*. Následne, v časti *Configuration* je potrebné vybrať najprv UPC UA server, na ktorý sa chceme pripojiť, a do poľa pre filtrovanie uzlov z adresného priestoru, je potrebné zadať FIO (nakoľko chceme pristupovať k premenným iba z tohto zoznamu). Akonáhle sa pripojíme na daný OPC UA server, všetky senzory a aktuátory namapujeme

(napárujeme) na premenné, ktoré boli získané zo zoznamu FIO (obrázok č. 149). Týmto bola nakonfigurovaná OPC UA komunikácia medzi softPLC a skladovým systémom vo Factory I/O.



Obrázok č. 148: Výber zariadenia [76]

Obrázok č. 149: Mapovanie senzorov a aktuátorov vo Factory IO na OPC UA premenné [76]

4.4.6 Implementácia komunikácie v Node-RED

Ďalším krokom v implementačnej časti je vytvorenie spojenia medzi virtuálnym skladovým systémom a mobilnou aplikáciou. Preto sme využili middleware Node-RED. Na napojenie na OPC UA server z Node-REDu sú potrebné uzly, ktoré podporujú takýto typ komunikácie. Na tento účel bolo potrebné nainštalovať knižnicu *node-red-contrib-opcua*. Na napojenie sa na MQTT broker (v našom prípade je to cloudový broker HiveMQ), sú tiež potrebné uzly pre MQTT protokol. Node-RED predvolene podporuje takýto typ komunikácie, keďže je častou voľbou pre IoT riešenia, a teda nebolo potrebné nič inštalovať.

V Node-RED editore máme jeden tok (flow), ktorý v sebe obsahuje dva typy pre-

pojenia: *OPC UA — MQTT* a *MQTT — OPC UA*.

KOMUNIKÁCIA OPC UA — MQTT

Implementácia komunikácie *OPC UA — MQTT* je znázornená na obrázku č. 150.

Na prijímanie údajov z OPC UA servera je využitý uzol *OPC UA Client* s nasledovnými nastaveniami:

- Endpoint — URL na OPC UA server (v našom prípade ide o *//localhost:4840*);
- Action — pre príjem dát z OPC UA servera ako akciu nastavíme Subscribe.

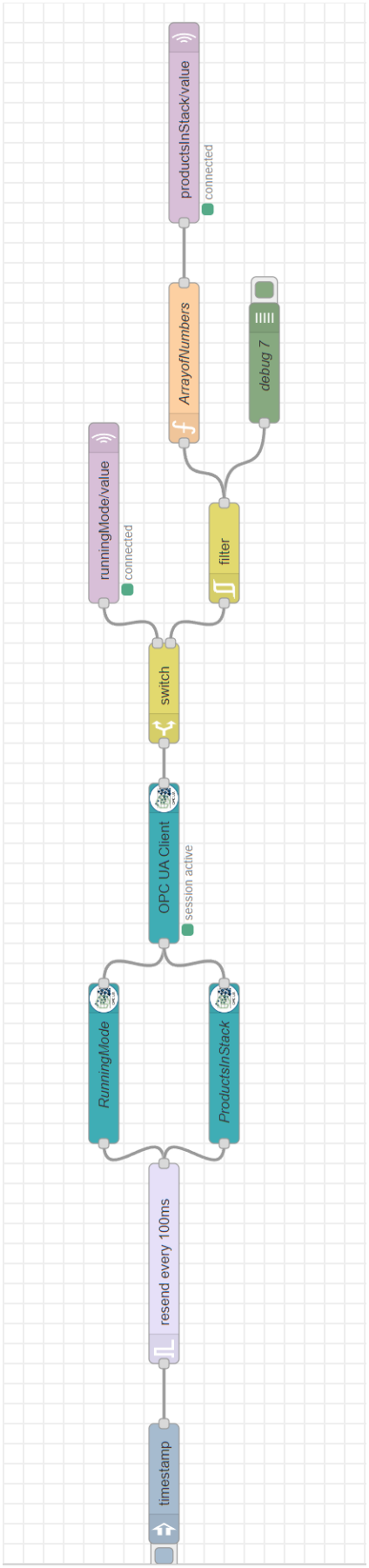
Vstupom do uzla *OPC UA klient* sú dva *OPC UA Item* uzly, ktoré predstavujú uzly/premenné, ku ktorým chceme pristúpiť z OPC UA servera. V tomto prípade, keďže potrebujeme informáciu o tom, či proces beží a aký je aktuálny stav v sklade, tak pristupujeme k premenným z OPC UA serveru *RunningMode* a *ProductsInStack*. *OPC UA Item* uzly majú tieto nastavenia:

- Item — adresa premennej, ktorá sa nachádza na OPC UA Serveri (pre premennú *RunningMode* je napríklad adresa: *ns=4;s=|var|CODESYS Control Win V3 x64.Application.ControlPou.RunningMode*)
- Type — dátový typ premennej (pre premennú *RunningMode* je dátový typ Boolean a pre *ProductsInStack* je typ UInt16)

Nakoľko bola zvolená v uzle *OPC UA client* akcia Subscribe, tak je potrebné nastaviť časový interval, ako často chceme prijímať údaje z OPC UA servera. V tomto prípade sme nastavili interval na 100 ms, aby boli údaje čo najaktuálnejšie. Následne výstup z uzla *OPC UA Client* zachytávame pomocou uzla *switch*, kde pre každý topic (topic je teda adresa premennej z OPC UA servera) nastavíme správny výstup, aby sme každú hodnotu, ktorú získame zo servera, vedeli zaslať na správny topic do MQTT brokera. Pre hodnotu premennej uzla *ProductsInStack* sme ešte predtým pridali uzol *filter*, ktorý bude posielať hodnoty na MQTT broker len v prípade, ak sa naozaj nejaká zmena v skladovom systéme udiala, čo šetrí dátové spojenie.

Hodnoty na MQTT broker zasielame pomocou uzlov *MQTT Publish*, ktoré majú nasledovné nastavenie:

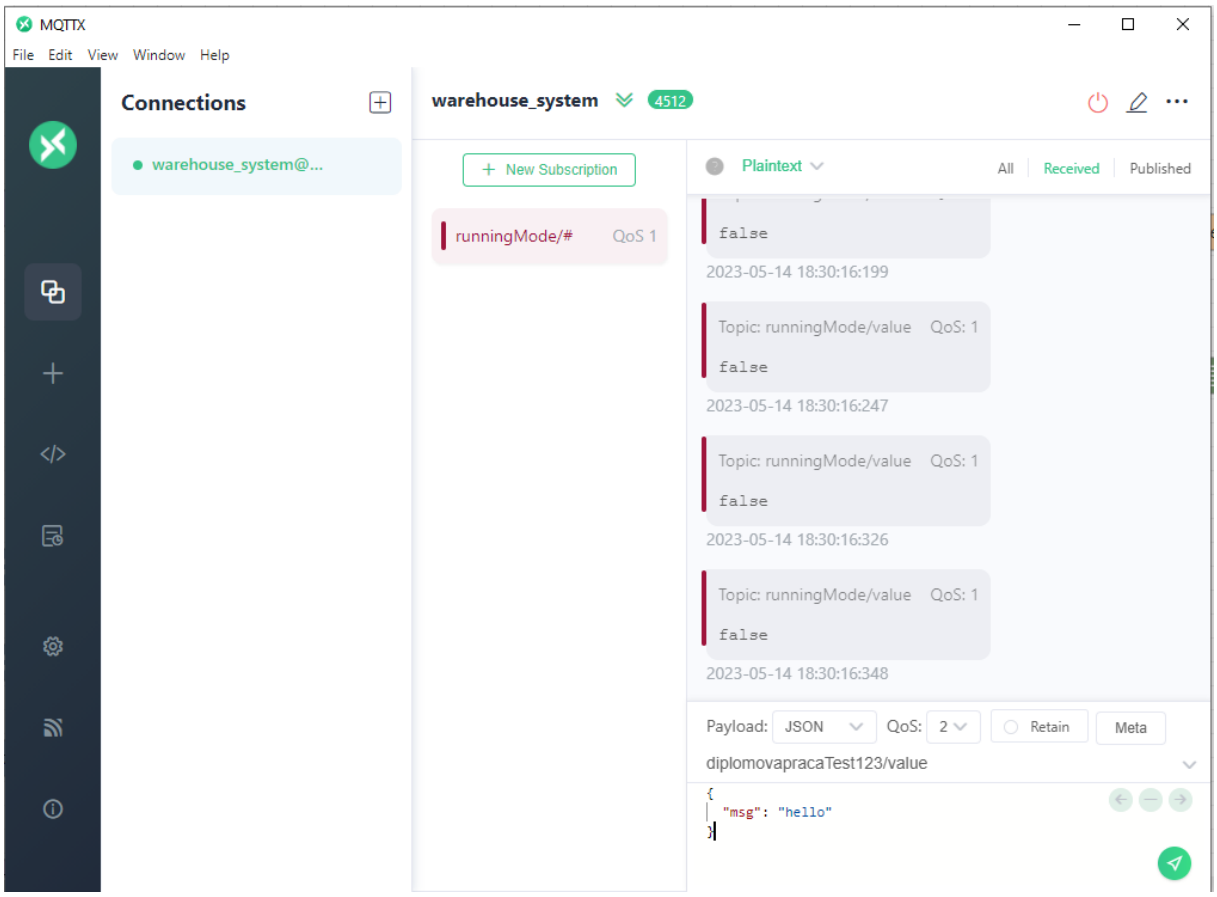
- Server — adresa servera/brokera HiveMQ (v našom prípade adresa je: *//broker.hivemq.com:1883*)
- Topic — pod akou témou (topicom) bude hodnota, ktorú posielame na broker (napr. pre hodnotu premennej *RunningMode* topic je: *runningMode/value*);



Obrázok č. 150: Implementácia komunikácie OPC UA — MQTT [76]

- QoS — nastavenie kvality služieb (pre oba uzly bolo nastaené QoS2, aby bolo isté, že hodnota bude doručená presne jeden raz);
- Security — tu bolo nastavené meno a heslo, aby dáta, ktoré posielame na broker, neboli verejné pre každého.

Pre uistenie, že sa hodnoty z OPC UA klienta zaslali na MQTT broker, bol urobený test pomocou softvéru MQTTX, kde bol vytvorený testovací MQTT klient, ktorý sa predplatil (subscribe) na topic *runningMode/value* pod daným menom a heslom. Z obrázka č. 151 je možné vidieť, že sa hodnoty naozaj poslali.

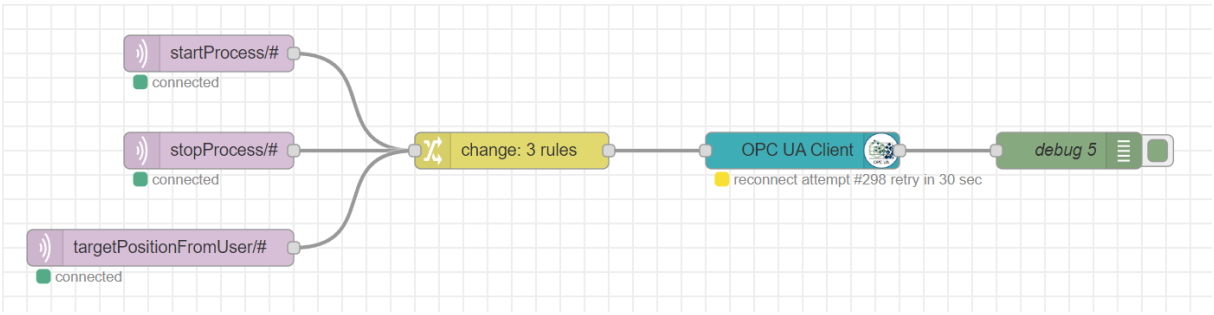


Obrázok č. 151: Testovanie komunikácie OPC UA — MQTT [76]

KOMUNIKÁCIA MQTT — OPC UA

Na posielanie dát z mobilnej aplikácie, využívame protokol MQTT s uzlom *MQTT Publish*. Dáta, ktoré operátor pošle z mobilnej aplikácie odchytávame v Node-RED pomocou MQTT uzla *Subscribe*. Hodnoty, ktoré posielame na OPC UA server, sú: *StartProcess* — na naštartovanie procesu, *StopProcess* — na zastavenie procesu a *targetPositionFromUser* — hodnota, ktorú operátor zvolil pre danú pozíciu výrobku v sklade. Ako je možné vidieť na obrázku č. 152, tak na začiatku máme tri *MQTT Subscribe* uzly, ktoré sa predplatia

(subscribe) na daný topic z MQTT brokera. Následne pomocou uzla *change* pre každý uzol správne nastavíme jeho topic. Aby sme totiž poslali údaje na server, potrebujeme vedieť adresu premennej, do ktorej hodnotu posielame. Akonáhle získame výstup z uzla *change*, presmerujeme ho na *OPC UA client*. V tomto prípade, keďže údaje zapisujeme, *OPC UA client* má ako Action nastavenú hodnotu WRITE.



Obrázok č. 152: Komunikácia MQTT — OPC UA [76]

4.4.7 Implementácia mobilnej aplikácie

Mobilná aplikácia bola realizovaná pomocou frameworku pre multiplatformový vývoj Ionic, o ktorom je uvedených viac informácií v predošlej časti tejto učebnice.

Na základe požiadaviek, ktoré boli definované a znázornené na obrázku č. 139, mobilná aplikácia obsahuje dve obrazovky, resp. dva komponenty: domovskú obrazovku (*HomeComponent*) a obrazovku WarehouseControl (*WarehouseControlComponent*).

Všetky technológie a zariadenia, ktoré boli využité pre vývoj mobilnej aplikácie, sme uviedli v tabuľkách č. 3, 4 a 5.

Tabuľka č. 3: Vývojové prostredia [76]

VÝVOJOVÉ PROSTREDIA	
Názov	Verzia
Visual Studio Code	1.78.2
Android Studio Dolphin	2021.3.1
Xcode	14.3

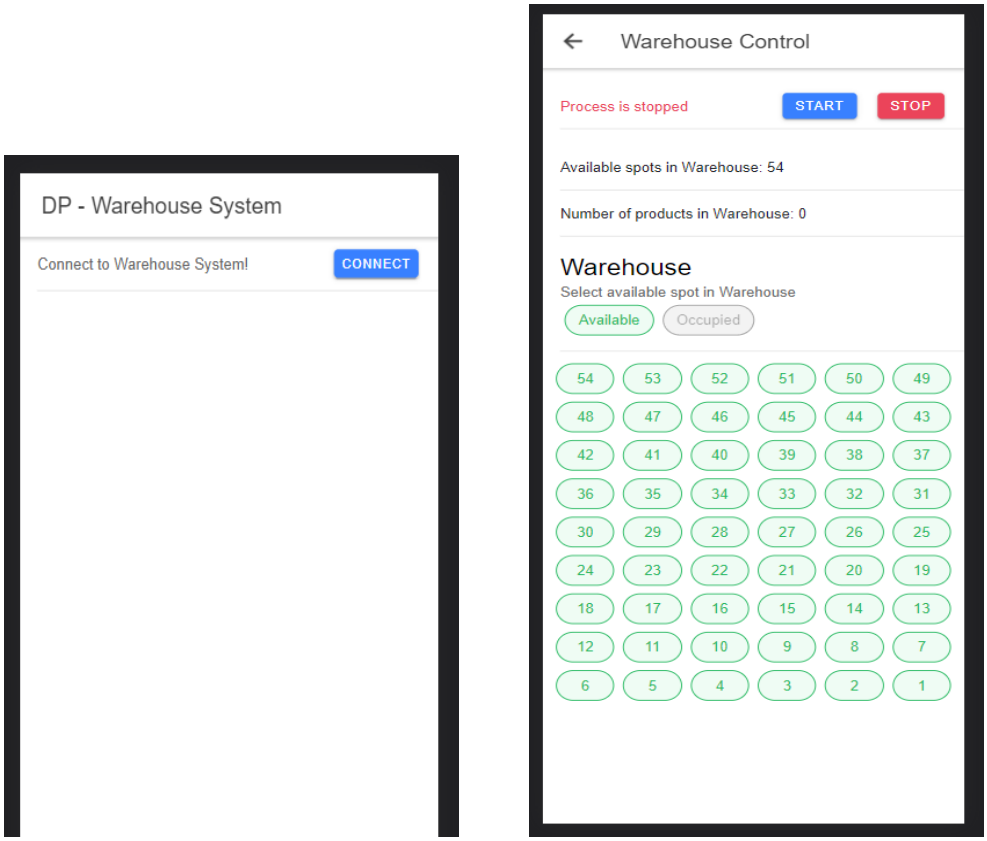
Tabuľka č. 4: Frameworky pre vývoj mobilnej aplikácie [76]

FRAMEWORK	
Názov	Verzia
Ionic	6.20.9
Angular	15.0.0

Tabuľka č. 5: Mobilné zariadenie pre vývoj mobilnej aplikácie [76]

MOBILNÉ ZARIADENIE		
Názov	OS	Opis
iPhone 7	**iOS 15.7.6**	fyzické zariadenie
Google Pixel 7	**Android 13**	fyzické zariadenie
Google Pixel 3a	**Android 13**	emulátor
iPhone 14 pro	**iOS 15.7.5**	emulátor

Domovská obrazovka (obrázok č. 153) sa načíta ako prvá a obsahuje iba jedno tlačidlo, pomocou ktorého sa operátor napojí na MQTT broker. Ak je konekcia úspešná, načíta sa obrazovka WarehouseControl.



(a) Domovská obrazovka (b) WarehouseControl obrazovka

Obrázok č. 153: Obrazovky v mobilnej aplikácii

Obrazovka WarehouseControl (obrázok č. 153 vpravo) obsahuje dve tlačidlá: *Start* a *Stop* a informáciu o tom, či proces beží. Ďalej tiež obsahuje informácie o počte voľných a obsadených miest v sklade. Ďalej obsahuje schematickú vizualizáciu skladového systému, kde má operátor možnosť si zvoliť miesto v sklade, kam chce výrobok uskladniť.

Nakoniec, má možnosť sa z MQTT brokera odpojiť pomocou tlačidla v ľavom hornom rohu. Ten ho zároveň vráti späť na domovskú obrazovku.

Schematická vizualizácia skladového systému bola vytvorená pomocou mriežky (*grid*), ktorá obsahuje 54 položiek. Položky sú vlastne voľne dostupné UI komponenty <*ion-chip*> obsahujúce rôzne možnosti štýlovania už vopred pripravené. My sme ich nastavili tak, aby sa dostupné miesta v sklade vyznačili zelenou farbou a obsadené miesta šedou farbou. Taktiež pre obsadené miesta bolo nastavené, aby boli deaktivované, teda aby sa na ne nedalo kliknúť.

IMPLEMENTÁCIA KOMUNIKÁCIE V MOBILNEJ APLIKÁCII

Na komunikáciu medzi mobilnou aplikáciou a Node-REDom, ako už bolo spomenuté, využívame protokol MQTT. Aby sme mohli nastaviť MQTT klienta v mobilnej aplikácii, bolo potrebné do Ionic projektu nainštalovať knižnicu *ngx-mqtt*. Výhodou tejto knižnice je, že ponúka metódy pre posielanie, prijímanie, napojenie a odpojenie sa z MQTT brokera. Metóda pre prijímanie dát z MQTT brokera má ako návratovú hodnotu Observable (pozorovateľ) typu *IMqttMessage*. Toto v princípe znamená: pozoruj správu z MQTT brokera a získaj hodnotu len vtedy, keď si odber predplatíš (vykonáš subscribe). Toto v opisovanom prípade zabezpečuje knižnica *RxJS*. Výhodou tohto prístupu k dátam je, že ich môžeme predtým, ako vykonáme zápis hodnôt do nejakej premennej, upraviť podľa toho, aký výstup chceme mať.

Pre lepšiu prehľadnosť, bola v rámci projektu vytvorená verejná trieda *MqttClientService*, ktorá obsahuje všetky metódy potrebné pre MQTT klienta. *Service* je špeciálny typ triedy, ktorý sa v Angulari bežne využíva na poskytovanie funkcionalít a dát v rámci aplikácie.

Najprv bol vytvorený súkromný objekt *connection*, ktorý obsahuje všetky nastavenia potrebné pre vytvorenie MQTT klienta (programový modul č. 8).

Na napojenie a odpojenie sa z MQTT brokera máme dostupné verejné metódy *connect*() a *disconnect*() (programový modul č. 9). Metóda *connect*() nám vracia hodnotu True alebo False v závislosti od toho, či sa nám podarilo napojiť sa na MQTT broker.

Metódu *connect*() voláme v metóde *establishConnection*(), ktorú máme v komponente *HomePage* (Domovská obrazovka). Vtedy, keď operátor stlačí tlačidlo Connect, zavolá sa metóda *establishConnection*(), ktorá následne zavolá metódu *connect*(). V prípade, že sa podarilo napojiť na MQTT broker, operátor bude presmerovaný na obrazovku Warehouse Control. V opačnom prípade sa zobrazí chybová hláška.

Ďalšou verejnou metódou je *observeMultiple*() (programový modul č. 10), ktorej vstupným argumentom je pole *topics*. Toto pole vlastne obsahuje dané témy (topics),

Programový modul č. 8 Objekt connection [76]

```
private connection = {
    hostname: 'broker.hivemq.com',
    port: 8000,
    path: '/mqtt',
    clean: true,
    connectTimeout: 4000,
    reconnectPeriod: 4000,
    username: 'meno',
    password: 'heslo',
  };
```

Programový modul č. 9 Metóda *connect*() a *disconnect*() [76]

```
connect(): boolean {
    try {
      this.mqttService?.connect(this.connection);
    } catch (error) {
      console.log('mqtt.connect error', error);
      return false;
    }
    return true;
  }

  disconnect(): void {
    this.mqttService.disconnect();
  }
```

ktoré chceme pozorovať z MQTT brokera HiveMQ. Akonáhle sa na danú tému zapíše nejaká hodnota v MQTT brokeri, vykonáme odber (subscribe) a získame hodnotu. Návratovou hodnotou tejto metódy je *Observable* typu pole *IMqttMessage*.

Programový modul č. 10 Metóda *observeMultiple*() [76]

```
observeMultiple(topics: string[]): Observable<IMqttMessage[]> {
    const observables = topics.map((topic) => this.mqttService.observe(topic));
    return combineLatest(observables);
  }
```

Metóda *observeMultiple*() je volaná v metóde *doSubscribe*() z komponentu *WarehouseControlComponent* (obrazovka Warehouse Control). Metóda *doSubscribe*() sa zavolá, akonáhle sa načíta obrazovka Warehouse Control, nakoľko hneď chceme získať údaje zo skladového systému. Vstupom do metódy *observeMultiple*() je pole tém (topics), čo v našom prípade sú témy *'runningMode/#'* a *'productsInStack/#'*. Akonáhle vykonáme odber (subcribe) na tieto témy, tak získame hodnoty o tom, či proces beží a aký je stav v sklade.

Nakoniec tu máme verejnú metódu *publishMessage*() (programový modul č. 11), ktorej vstupným argumentom je objekt *publishPayload* s nasledovnými členmi: *topic* — pod akou témou (topic) chceme hodnotu poslať na MQTT broker, *payload* — užitočná hodnota (obsah premennej) pre danú tému a *qos* na nastavenie kvality služby.

Programový modul č. 11 Metóda *publishMessage*() [76]

```
publishMessage(publishPayload: {topic: string; payload: string; qos: number;}): void {
    const { topic, qos, payload } = publishPayload;
    console.log(publishPayload);
    this.mqttService.unsafePublish(topic, payload, { qos } as IPublishOptions);
  }
```

Táto metóda je volaná v metódach *startProcess*(), *stopProcess*() a *sendValue*(), ktoré máme v komponente *WarehouseControlComponent* (obrazovka Warehouse Control).

Akonáhle operátor zatlačí tlačidlo Start, volá sa metóda *startProcess*(), ktorá následne zavolá metódu *publishMessage*(). Ako parameter do metódy *publishMessage*() posielame objekt *publishStartProcess* (programový modul č. 12).

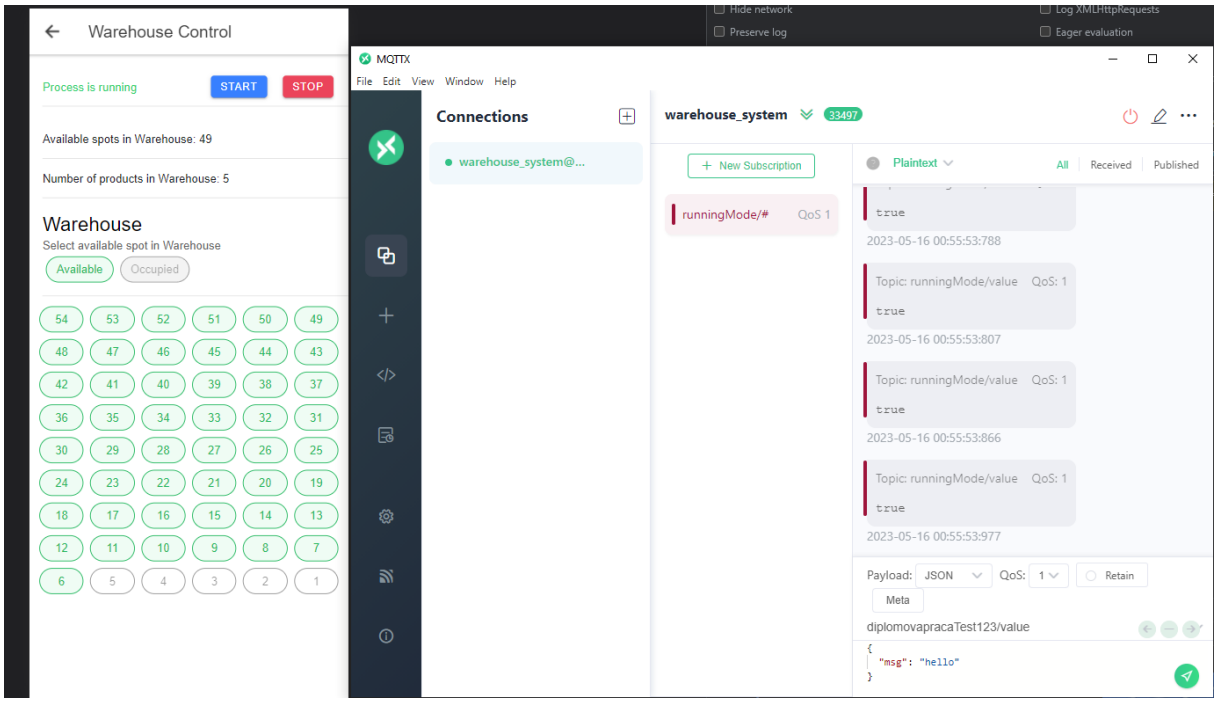
Rovnakým spôsobom pracujeme aj pre metódy *stopProcess*() a *sendValue*(), pričom posielame rôzne parametre pre príslušné témy (topics) v MQTT brokeri.

Na uistenie, že sa hodnoty z MQTT klienta z mobilnej aplikácie poslali na MQTT broker, otestovali sme spojenie opäť pomocou softvéru MQTTX, kde máme rovnakého

Programový modul č. 12 Objekt *publishStartProcess* [76]

```
publishStartProcess = {
    topic: 'startProcess/value',
    qos: 2,
    payload: 'true',
  };
```

testovacieho MQTT klienta (obrázok č. 151), ktorý sa prihlásil na odber (subscribe) na tému *runningMode/value* pod daným menom a heslom. Z obrázka č. 154 vidíme z mobilnej aplikácie (vľavo) informáciu, že proces beží a taktiež aj klient v MQTTX (vpravo), zobrazuje hodnotu z témy *runningMode/value* ako TRUE.

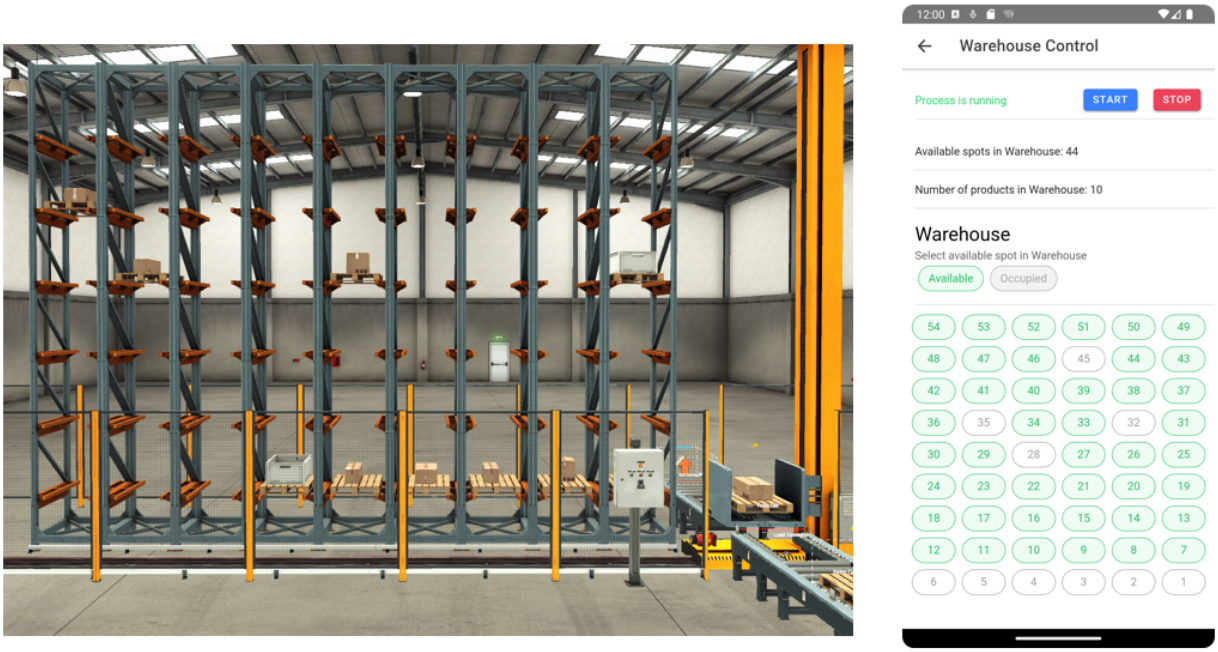


Obrázok č. 154: Testovanie komunikácie MQTT klient — MQTT broker [76]

4.4.8 Zhrnutie prípadovej štúdie

Akonáhle sa podarilo prepojiť komunikácie medzi jednotlivými technológiami, celý systém sme spustili, nechali bežať a sledovali výstupy. Na obrázku č. 155 vidíme konečný výstup prepojenia virtuálneho skladového systému vo Factory I/O a mobilnou aplikáciou realizovanou pomocou frameworku Ionic. Taktiež je vidno aj to, že oba aplikačné systémy navzájom medzi sebou komunikujú a vymieňajú aktuálne dáta.

Nakoľko bolo žiadané, aby konečný výsledok bol dostupný aj pre iné platformy mobilných zariadení, tak vďaka Ionicu a Capacitoru celý projekt je možné konfigurovať tak, aby sme mohli vytvoriť mobilné aplikácie pre platformy Android a iOS. Táto časť je re-



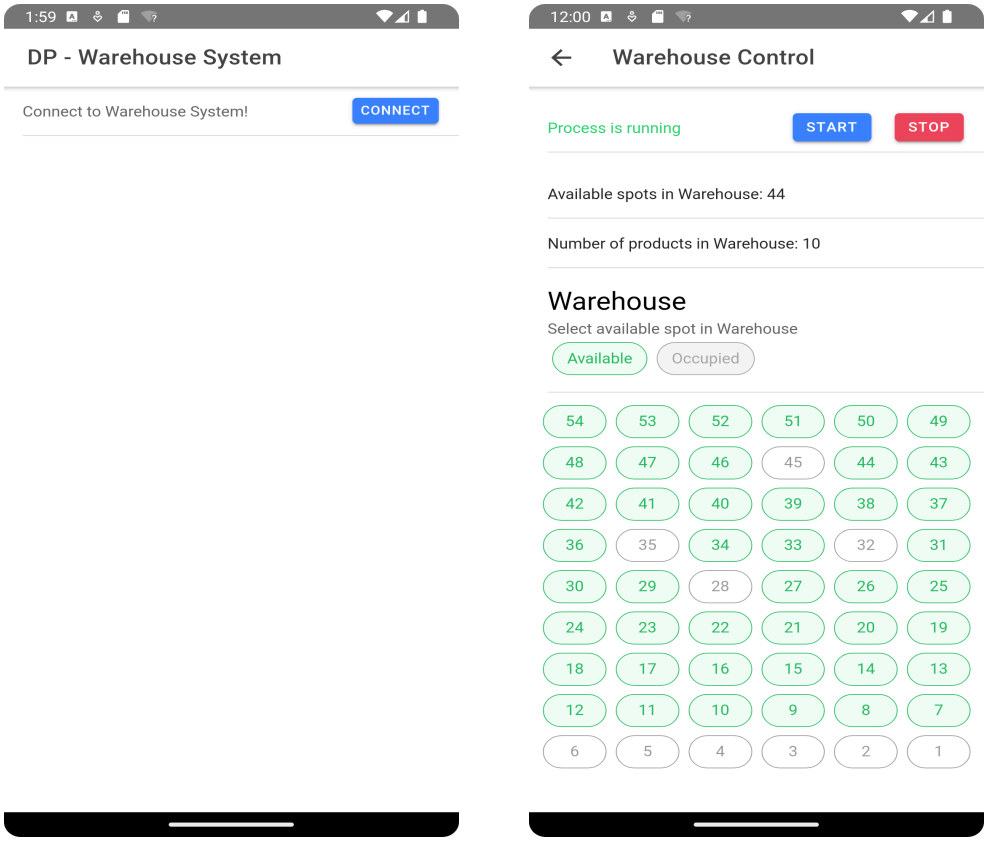
Obrázok č. 155: Prepojenie skladového systému s mobilnou aplikáciou [76]

latívne jednoduchá, nakoľko stačí iba spustiť nasledovné príkazy:

- *ionic capacitor copy android* — príkaz ktorý vytvorí verziu projektu pre platformu Android;
- *ionic capacitor copy ios* — príkaz ktorý vytvorí verziu projektu pre platformu iOS.

Projekt pre platformu Android sme spustili v Android Studiu, ktorý nám vygeneroval súbor *apk*. Funkcionalitu mobilnej aplikácie sme najprv otestovali na Android emulátore v Android Studiu (obrázok 156). Následne sme si aplikáciu nainštalovali aj na fyzické Android zariadenie a otestovali jej funkcionalitu.

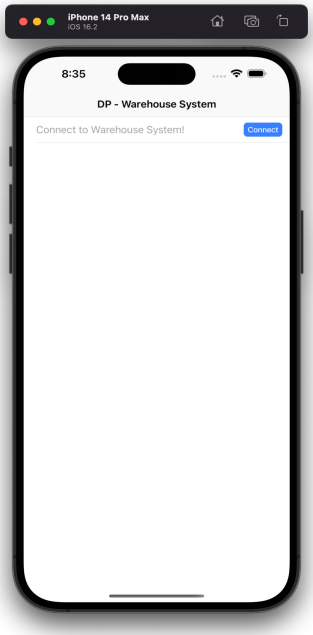
Projekt pre platformu iOS sme spustili v Xcode, ktorý vygeneroval súbor typu *ipa*. Funkcionalita pre iOS verziu bola najprv otestovaná na iOS emulátore v Xcode a potom aj na fyzickom iOS zariadení (obrázok č. 157).



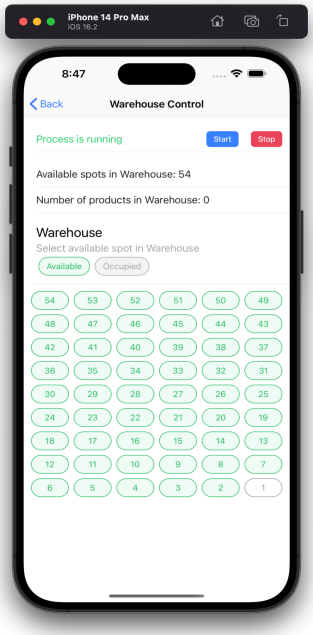
(a) Domovská obrazovka

(b) WarehouseControl obrazovka

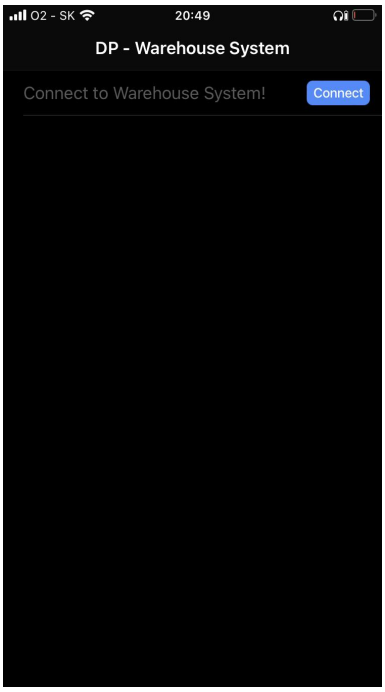
Obrázok č. 156: Obrazovky v mobilnej aplikácie v Android emulátore [76]



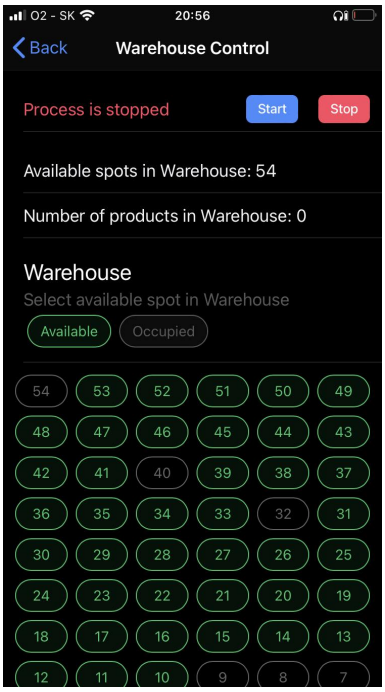
(a) emulátor



(b) emulátor



(c) fyzické zariadenie



(d) fyzické zariadenie

Obrázok č. 157: Obrazovky na iOS zariadení [76]

Predstavené riešenie je ďalej možné rozvíjať a vylepšovať. Napríklad, v skladovom systéme by mohla byť pridaná funkcionalita pre triedenie výrobkov, ktoré sa budú ukladať do skladu. V rámci tohto by sa mohla pridať funkcionalita v mobilnej aplikácii, aby si operátor mohol vybrať, či chce výrobky zoraďovať a zvoliť si možnosť, podľa akého

kľúča by ich triedil.

Taktiež sa v niektorých prípadoch v skladovom systéme môže stať, že výrobok, ktorý sa už v sklade nachádza, zo skladového miesta vypadne. Toto by sa dalo vylepšiť tým, že by sa pridal ďalší senzor alebo kamera, ktorá by snímala a detegovala spomínané situácie a oznamovala operátorovi v mobilnej aplikácii prípadnú poruchu.

Ďalším možným vylepšením je zakomponovanie databázového systému. Všetky informácie o skladovom systéme by sa ukladali do databázy, aby operátor mal všetky informácie k dispozícii kedykoľvek.

Videá k tejto edukačnej prípadovej štúdii je možné nájsť na adrese:

- `https://bit.ly/47S5L8Q`

Ďalšie informácie, návody a programové projekty je možné nájsť na e-learningovej webovej stránke `https://elearning.mechatronika.cool/?p=8350` .

Zoznam použitej literatúry

[1] Asad, U., Khan, M., Khalid, A., a Lughmani, W. A. Human-Centric Digital Twins in Industry: A Comprehensive Review of Enabling Technologies and Implementation Strategies. *Sensors 23*, 8 (2023).

[2] Ashton, K., et al. That 'Internet of Things' thing. *RFID journal 22*, 7 (2009), 97–114.

[3] Benešová, A., a Tupa, J. Requirements for education and qualification of people in industry 4.0. *Procedia Manufacturing 11* (2017), 2195 – 2202. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. https://doi.org/10.1016/j.promfg.2017.07.366.

[4] Beňo, L. *Moderné metódy ovládania mechatronických zariadení s využitím hlasových povelov založené na technológii Edge computingy*. Dizertačná práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2022. 106 s.

[5] Beňo, L., Pribiš, R., a Drahoš, P. Edge Container for Speech Recognition. *Electronics 10*, 19 (2021). DOI: 10.3390/electronics10192420.

[6] Bourke, K. *What is CODESYS and Why is it Important?* Real Pars. [online]. [cit. 2023-04-26]. Dostupné z: `https://www.motioncontroltips.com/instruction-lists-ils-plc-programming/`.

[7] Brecko, A., Kajáti, E., a Papcun, P. Industry 5.0 – technológie: bio-inšpirované technológie a inteligentné materiály. *ATP Journal 02/2022* (2022).

[8] Brecko, A., Kajáti, E., a Zolotová, I. Industry 5.0 – technológie: interakcie medzi človekom a strojom. *ATP Journal 01/2022* (2022), 40–41.

[9] Briš, L. *Využitie formalizmov Petriho sietí v riadení laboratórnych systémov*. Diplomová práca (*Vedúci práce: Alena Kozáková; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2016. 58 s.

[10] Bucsai, S. *Ovládanie a monitorovanie IoT zariadení s využitím zmiešanej reality*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2019.

[11] Bucsai, S., Kučera, E., Haffner, O., a Drahoš, P. Control and Monitoring of IoT Devices Using Mixed Reality Developed by Unity Engine. In *2020 Cybernetics & Informatics (K&I)* (2020), pp. 1–8.

[12] Burger, A., Koziolek, H., Rückert, J., Platenius-Mohr, M., a Stomberg, G. Bottleneck Identification and Performance Modeling of OPC UA Communication Models. In *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering* (Apr. 2019), ICPE '19, ACM. DOI: 10.1145/3297663.3309670.

[13] Cartwright, A. *OPC-UA: the Flow of Data*. [online]. [cit. 2024-01-20]. Dostupné z: `https://medium.com/ai-build-techblog/opc-ua-the-flow-of-data-7c3e5c870a4c`.

[14] Cavalieri, S., Salafia, M. G., a Scroppo, M. S. Integrating OPC UA with web technologies to enhance interoperability. *Computer Standards & Interfaces 61* (Jan. 2019), 45–64. DOI: 10.1016/j.csi.2018.04.004.

[15] Chaudhry, T., Juneja, A., a Rastogi, S. AR Foundation for Augmented Reality in Unity. *International Journal of Advances in Engineering and Management 3*, 1 (2021), 1–7.

[16] Ciolacu, M. I., Svasta, P., Berg, W., a Popp, H. Education 4.0 for Tall Thin Engineer in a Data Driven Society. *2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)* (2017), 432–437.

[17] Coito, T., Martins, M. S., Viegas, J. L., Firme, B., Figueiredo, J., Vieira, S. M., a Sousa, J. M. A Middleware Platform for Intelligent Automation: An Industrial Prototype Implementation. *Computers in Industry 123* (Dec. 2020), 103329. DOI: 10.1016/j.compind.2020.103329.

[18] Croatti, A., a Ricci, A. Towards the Web of Augmented Things. In *Software Architecture Workshops (ICSAW), 2017 IEEE International Conference on* (2017), IEEE, pp. 80–87.

[19] Dean, James. *First-of-its-kind 3D-printed home blends concrete, wood*. [online]. [cit. 2024-01-27]. Dostupné z: `https://news.cornell.edu/stories/2022/09/first-its-kind-3d-printed-home-blends-concrete-wood`.

[20] Drath, R., Mosch, C., Hoppe, S., Faath, A., Barnstedt, E., Fiebiger, B., a Schlögl, W. Diskussionspapier–interoperabilität mit der verwaltungsschale, opc ua und automationml. *Technical Report. AutomationML eV and Industrial Digital Twin Association (IDTA) and OPC Foundation and VDMA* (2023).

[21] Dvořák, J. *Integrace měřičů spotřeby energií do SCADA systému, zpracování a vyhodnocení dat*. Diplomová práca (*Vedúci práce: Jan Holub*). Praha: ČVUT Fakulta elektrotechnická, 2017. 68 s.

[22] European Commission - Directorate General for Research and Innovation. *Industry 5.0: towards a sustainable, human centric and resilient European industry.* Publications Office, 2021. DOI: 10.2777/308407.

[23] Fischertechnik. *Indexed Line with two Machining Stations 24V - Simulation*. [online]. [cit. 2021-04-07]. Dostupné z: https://www.fischertechnik.de/en/products/simulating/training-models/96790-sim-indexed-line-with-two-machining-stations-24v-simulation.

[24] Fischertechnik. *Punching Machine with Conveyor Belt 24v*. [online]. [cit. 2023-04-29]. Dostupné z: https://www.fischertechnik.biz/punching-machine-with-conveyor-belt-24v.

[25] Fortunato, D., a Bernardino, J. Progressive Web Apps: An Alternative to the Native Mobile Apps. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)* (2018), IEEE, pp. 1–6.

[26] Gallasch, A. ThingWorx–Plattform zur Integration herausfordernder Anforderungen auf dem Shopfloor. *Produktions-und Verfügbarkeits-optimierung mit Smart Data Ansätzen* (2018), 83–92.

[27] Grenčíková, A., Kordoš, M., a Navickas, V. The Impact of Industry 4.0 on Education Contents. *Business: Theory and Practice 22*, 1 (Jan. 2021), 29–38. DOI: 10.3846/btp.2021.13166.

[28] Guinard, D., Trifa, V., Pham, T., a Liechti, O. Towards Physical Mashups in the Web of Things. In *Networked Sensing Systems (INSS), 2009 Sixth International Conference on* (2009), IEEE, pp. 1–4.

[29] Habib, K., Saad, M. H. M., Hussain, A., Sarker, M. R., a Alaghbari, K. A. An Aggregated Data Integration Approach to the Web and Cloud Platforms through a Modular REST-Based OPC UA Middleware. *Sensors 22*, 5 (2022). DOI: 10.3390/s22051952.

[30] Haffner, O., a Kučera, E. Multiplatform Mobile Application for Identification and Localization of Objects in Space. In *2020 Cybernetics & Informatics (K&I)* (2020), pp. 1–9. DOI: 10.1109/KI48306.2020.9039849.

[31] Halliwell, R. *Programmable Logic Controllers*. op-tec. [online]. [cit. 2023-05-17]. Dostupné z: https://op-tec.co.uk/knowledge/using-programmable-logic-controllers/.

[32] Hrúz, B., a Mrafko, L. *Modelovanie a riadenie diskrétnych udalostných systémov: s využitím Petriho sietí a iných nástrojov*. Bratislava: Vydavateľstvo STU, 2003. ISBN: 80-227-1883-1.

[33] Huba, M., a Kozák, Š. From e-Learning to Industry 4.0. In *2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)* (Nov 2016), pp. 103–108. DOI: DOI: 10.1109/ICETA.2016.7802083.

[34] Imtiaz, J., a Jasperneite, J. Scalability of OPC-UA down to the chip level enables "Internet of Things". In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)* (July 2013), IEEE. DOI: 10.1109/indin.2013.6622935.

[35] Kajáti, E., a Zolotová, I. Industry 5.0 – revolúcia alebo evolúcia? *ATP Journal 12/2021* (2021), 36–37.

[36] Kolektív Advantech. *Quick Guide to OPC UA and Advantech IT/OT Convergent Solutions*. [online]. [cit. 2023-05-04]. Dostupné z: `https://page.advantech.com/en/global/industrial-automation/industrial-io/opc-ua`.

[37] Kolektív Automatizácia 365. *Ktoré PLC programovacie jazyky najviac používame?* [online]. [cit. 2022-05-17]. Dostupné z: `https://www.automatizacia365.sk/2021/04/13/ktore-plc-programovacie-jazyky-najcastejsie-pouzivame`.

[38] Kolektív autorov www.opto22.com. Ako preklenúť medzeru medzi IT a OT. *ATP Journal 1/2017* (2017), 32–33.

[39] Kolektív Firmata. *Firmata firmware for Arduino. GitHub*. 2016. [online]. [cit. 2020-08-20]. Dostupné z: `https://github.com/firmata/arduino`.

[40] Kolektív HiveMQ. *MQTT Essentials*. [online]. [cit. 2020-06-05]. Dostupné z: `https://www.hivemq.com/tags/mqtt-essentials/`.

[41] Kolektív Node-RED. *About Node-RED*. [online]. [cit. 2020-06-10]. Dostupné z: `https://nodered.org/about/`.

[42] Kolektív Node-RED. *Node-RED User Guide*. [online]. [cit. 2020-06-10]. Dostupné z: `https://nodered.org/docs/user-guide/`.

[43] Kolektív OpenPLC Project. *Modbus Address Mapping*. [online]. [cit. 2021-02-01]. Dostupné z: `https://www.openplcproject.com/reference/modbus/`.

[44] Kolektív Produktion 2030. *Ingenjör4.0*. [online]. [cit. 2021-02-07]. Dostupné z: `https://produktion2030.se/en/ingenjor-4-0`.

[45] Kolektív randed.com. *Information Technologies (IT) Vs Operational Technologies (OT). Webranded*. [online]. [cit. 2020-08-06]. Dostupné z: https://randed.com/information-technologies-it-vs-operational-technologies-ot/?lang=en.

[46] Kolektív Reality Reflection. *Virtual, Augmented, Mixed-Reality; What is these all about?. Medium.com*. [online]. [cit. 2020-06-02]. Dostupné z: https://medium.com/@realityreflection/vr-virtual-augmented-mixed-reality-what-is-these-all-about-25762bfda62a.

[47] Kolektív The Foundry. *VR? AR? MR? Sorry, I'm confused. The Foundry*. [online]. [cit. 2020-06-02]. Dostupné z: https://www.foundry.com/insights/vr-ar-mr/vr-mr-ar-confused.

[48] Kolektív Úradu podpredsedu vlády Slovenskej republiky pre investície a informatizáciu. *Cloud computing. Informatizácia - Úrad podpredsedu vlády Slovenskej republiky pre investície a informatizáciu*. [online]. [cit. 2020-06-01]. Dostupné z: http://www.informatizacia.sk/cloud-computing/22841s.

[49] Kostoláni, M., Murín, J., a Kozák, Š. Intelligent Predictive Maintenance Control Using Augmented Reality. In *2019 22nd International Conference on Process Control (PC19)* (2019), pp. 131–135. DOI: 10.1109/PC.2019.8815042.

[50] Kozák, Š., a Hypiusová, M. *Moderné metódy a algoritmy riadenia*. Bratislava: Slovenská chemická knižnica, 2016. ISBN: 978-80-89597-44-4.

[51] Kozák, Š., Ružický, E., Štefanovič, J., a Schindler, F. Research and Education for Industry 4.0: Present Development. In *2018 Cybernetics & Informatics (K&I)* (Feb 2018), pp. 1–7.

[52] Kučera, E., Haffner, O., Drahoš, P., Cigánek, J., Leskovský, R., a Štefanovič, J. New Software Tool for Modeling and Control of Discrete-Event and Hybrid Systems Using Timed Interpreted Petri Nets. *Applied Sciences 10*, 15 (2020), 5027. DOI: 10.3390/app10155027.

[53] KUNBUS GmbH. *Open Source IPC based on Raspberry Pi. Revolution Pi*. [online]. [cit. 2020-08-06]. Dostupné z: https://revolution.kunbus.com/.

[54] Kučera, E. *Využitie Petriho sietí na modelovanie a riadenie skladových systémov*. Diplomová práca (*Vedúci práce: Leo Mrafko; Konzultant: Branislav Hrúz*). Bratislava: FEI STU, 2013. 81 s.

[55] Kučera, E. *Modelovanie a riadenie hybridných systémov s využitím Petriho sietí vyšších úrovní*. Dizertačná práca (*Vedúci práce: Štefan Kozák*). Bratislava: FEI STU, 2016. 121 s.

[56] Kučera, E. *Digitálne technológie a systémy pre Industry 4.0*. Habilitačná práca. Bratislava: FEI STU, 2020. 224 s.

[57] Kučera, E. *Internet of Things - slajdy z prednášok Virtuálna a zmiešaná realita pre Industry 4.0*. [online]. [cit. 2020-06-01]. Dostupné z: `http://vzri.mechatronika.cool/sites/default/files/Prednaska%2010%20IoT.pdf`.

[58] Kučera, E. *Prehľad cloudových služieb Microsoft Azure - slajdy z prednášok Virtuálna a zmiešaná realita pre Industry 4.0*. [online]. [cit. 2020-06-01]. Dostupné z: `http://vzri.mechatronika.cool/sites/default/files/Prednaska%208%20Azure.pdf`.

[59] Kučera, E. *Virtuálna, rozšírená a zmiešaná realita - slajdy z prednášok Virtuálna a zmiešaná realita pre Industry 4.0*. [online]. [cit. 2020-06-01]. Dostupné z: `http://vzri.mechatronika.cool/sites/default/files/Prednaska%207%20VR.pdf`.

[60] Kučera, E., Haffner, O., Drahoš, P., a Kozáková, A. Modeling and Control of Discrete Event and Hybrid Systems Using Petri Nets and OPC Unified Architecture. *IEEE Access 10* (2022), 120735–120751. DOI: 10.1109/ACCESS.2022.3222828.

[61] Kučera, E., Haffner, O., Drahoš, P., Leskovský, R., a Cigánek, J. PetriNet Editor + PetriNet Engine: New Software Tool For Modelling and Control of Discrete Event Systems Using Petri Nets and Code Generation. *Applied Sciences 10*, 21 (2020). DOI: 10.3390/app10217662.

[62] Képešiová, Z. *Monitorovanie a riadenie mechatronických systémov s využitím digitálneho dvojčaťa a zmiešanej reality*. Písomná práca k dizertačnej skúške (*Vedúci práce: Danica Rosinová; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2019.

[63] Leskovský, R. *Moderné metódy ovládania a diagnostiky mechatronických zariadení s využitím IoT a zmiešanej reality*. Písomná práca k dizertačnej skúške (*Vedúci práce: Danica Rosinová; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2020.

[64] Lewicki, P. *Controlling lights with the Hololens and Internet of Things*. htmlfusion. [online]. [cit. 2019-04-08]. Dostupné z: `https://tinyurl.com/jgfbso2`.

[65] Liang, Q., a Li, L. The Study of Soft PLC Running System. *Procedia Engineering 15* (2011), 1234–1238. CEIS 2011. DOI: https://doi.org/10.1016/j.proeng.2011.08.228.

[66] Lopatniková, S. *Modelovanie a riadenie udalostného systému s využitím PLC a cloudového riešenia*. Diplomová práca (*Vedúci práce: Oto Haffner; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2022. 81 s.

[67] Mahnke, W., Leitner, S.-H., a Damm, M. *OPC Unified Architecture*. Springer Science & Business Media, 2009.

[68] Marton, M. *Riadenie udalostných systémov s využitím Petriho sietí a OPC UA*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2022. 44 s.

[69] Miksad, M. *Prepojenie open-source PLC s počítačom generovanou realitou*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2023.

[70] Milgram, P., a Kishino, F. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems 77*, 12 (1994), 1321–1329.

[71] OpenBCI. *Ultracortex "Mark IV" EEG Headset*. [online]. [cit. 2024-01-27]. Dostupné z: `https://shop.openbci.com/collections/frontpage/products/ultracortex-mark-iv`.

[72] Organizácia spojených národov (OSN). *17 cieľov udržateľného rozvoja*. [online]. [cit. 2024-01-27]. Dostupné z: `https://unis.unvienna.org/unis/sk/topics/sustainable_development_goals.html`.

[73] Pajpach, M., Haffner, O., Kučera, E., a Drahoš, P. Low-Cost Education Kit for Teaching Basic Skills for Industry 4.0 Using Deep-Learning in Quality Control Tasks. *Electronics 11*, 2 (2022). DOI: 10.3390/electronics11020230.

[74] Pajpach, M., Pribiš, R., Drahoš, P., Kučera, E., a Haffner, O. Design of an Educational-development Platform for Digital Twins using the Interoperability of the OPC UA Standard and Industry 4.0 Components. In *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)* (2023), pp. 1–6. DOI: 10.1109/ICECCME57830.2023.10252941.

[75] Papcun, P., Mičko, K., a Kajáti, E. Industry 5.0 – technológie: bezpečný prenos, ukladanie a analýza údajov. *ATP Journal 04/2022* (2022), 52–53.

[76] Pavlovič, D. *Riadenie udalostného systému s využitím PLC a mobilnej aplikácie*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2023.

[77] Pohanka, P. *Internet věcí*. [online]. [cit. 2020-06-01]. Dostupné z: `http://i2ot.eu/internet-of-things/`.

[78] Popjáková, D., a Mintálová, T. Priemysel 4.0, čo mu predchádzalo a čo ho charakterizuje - geografické súvislosti. *Acta Geographica Universitatis Comenianae 63*, 2 (2019), 173–192.

[79] Pribiš, R., Beňo, L., a Drahoš, P. Asset Administration Shell Design Methodology Using Embedded OPC Unified Architecture Server. *Electronics 10*, 20 (2021). DOI: 10.3390/electronics10202520.

[80] Rambach, J., Pagani, A., Stricker, D., Aleksy, M., Schmitt, J., Langfinger, M., Schneider, M., Schotten, H., Malignaggi, A., Ko, M., et al. Augmented Things: Enhancing AR Applications leveraging the Internet of Things and Universal 3D Object Tracking. In *IEEE International Conference on Industrial Technology (ICIT)* (2017), vol. 22, p. 25.

[81] Riesz, M. *PNEditor - a Petri Net editor. PNEditor*. 2014. [online]. [cit. 2016-04-24]. Dostupné z: `http://www.pneditor.org/`.

[82] Romero, D., Bernus, P., Noran, O., Stahre, J., a Fast-Berglund, Å. The Operator 4.0: Human Cyber-Physical Systems & Adaptive Automation Towards Human-Automation Symbiosis Work Systems. In *Advances in Production Management Systems. Initiatives for a Sustainable World* (Cham, 2016), I. Nääs, O. Vendrametto, J. Mendes Reis, R. F. Gonçalves, M. T. Silva, G. von Cieminski, a D. Kiritsis, Eds., Springer International Publishing, pp. 677–686.

[83] Stark, E. *Moderné metódy ovládania a monitorovania mechatronických systémov s využitím počítačom generovanej reality*. Dizertačná práca (*Vedúci práce: Peter Drahoš; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2019. 120 s.

[84] Stark, E., Kučera, E., Haffner, O., Drahoš, P., a Leskovský, R. Using Augmented Reality and Internet of Things for Control and Monitoring of Mechatronic Devices. *Electronics 9*, 8 (2020), 1272.

[85] Sterling, I., a Swaroop, P. *Control with your smart devices by staring and gesturing*. Arduino. [online]. [cit. 2019-04-08]. Dostupné z: `https://blog.arduino.cc/2016/07/26/control-with-your-smart-devices-by-staring-and-gesturing/`.

[86] Tiegelkamp, M., a John, K.-H. *IEC 61131-3: Programming industrial automation systems*, vol. 166. Springer, 2010.

[87] Trifa, V., Guinard, D., a Carrera, D. *Web Thing Model*. W3C. [online]. [cit. 2020-04-08]. Dostupné z: `http://model.webofthings.io/`.

[88] UniPi.technology. *Automatizace Jednoduše. Unipi*. [online]. [cit. 2020-08-06]. Dostupné z: https://www.unipi.technology/cs/.

[89] Unity Technologies. *Create a marker-based AR app*. [online]. [cit. 2023-05-06]. Dostupné z: https://learn.unity.com/project/create-a-marker-based-ar-app.

[90] Viganò, G. P. *M2MQTT for Unity*. GitHub. [online]. [cit. 2023-05-06]. Dostupné z: https://github.com/gpvigano/M2MqttUnity.

[91] Voráčová, v., Pěnička, M., a Veselý, J. *Úvod do modelování procesů Petriho sítěmi*. Vyd. 1. Praha: Česká technika - nakladatelství ČVUT, 2008, 126 s. ISBN 978-80-01-03979-3.

[92] Wipro. *Edge Computing - Understanding the User Experience*. [online]. [cit. 2024-01-11]. Dostupné z: https://www.wipro.com/infrastructure/edge-computing-understanding-the-user-experience/.

[93] Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L., a Zhou, W. A Comparative Study of Containers and Virtual Machines in Big Data Environment. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)* (2018), pp. 178–185. DOI: 10.1109/CLOUD.2018.00030.

[94] Zolotová, I., Kajáti, E., a Pomšár, L. Industry 5.0 – koncept, technológie, ciele. *ATP Journal 11/2021* (2021), 42–43.

[95] Český institut informatiky, robotiky a kybernetiky ČVUT. *Testbed pro Průmysl 4.0*. České vysoké učení technické v Praze. [online]. [cit. 2020-06-03]. Dostupné z: https://www.ciirc.cvut.cz/cs/teams-labs/testbed/.

[96] Češek, P. *Riadenie DEDS pomocou mikrokontrolérov s využitím Petriho sietí*. Diplomová práca (*Vedúci práce: Alena Kozáková; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2016.

[97] Čížek, J. *Počítačový master/slave je společensky nekorektní. Připomíná otroctví, a proto zmizí*. [online]. [cit. 2024-01-20]. Dostupné z: https://www.zive.cz/clanky/pocitacovy-master/slave-je-spolecensky-nekorektni-pripomina-otroctvi-a-proto-zmizi/sc-3-a-204378/default.aspx.

[98] Štetáková, M. *Riadenie udalostného systému s využitím PLC a webovej aplikácie*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2023.

[99] Štěpánek, M. *Multiplatformová aplikácie pre získavanie dát o lokalitách s bezbariérovým prístupom*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2018. 40 s.

[100] Žemla, F., Cigánek, J., Rosinová, D., Kučera, E., a Haffner, O. Smart Platform for Monitoring and Control of Discrete Event System in Industry 4.0 Concept. *Applied Sciences 13*, 19 (2023). DOI: 10.3390/app131910697.

doc. Ing. Erik Kučera, PhD.

DIGITÁLNE A INTELIGENTNÉ TECHNOLÓGIE PRE INDUSTRY 4.0

Vydala Slovenská technická univerzita v Bratislave vo Vydavateľstve SPEKTRUM STU, Bratislava, Vazovova 5, v roku 2024.

Edícia vysokoškolských učebníc

Rozsah 283 strán, 213 obrázkov, 7 tabuliek, 22,654 AH, 22,942 VH,
1. vydanie, edičné číslo 6199.

85 – 217 – 2024

ISBN 978-80-227-5430-9
DOI: 10.61544/GYSJ2849